Mainak Jas

Real-time machine learning of MEG: Decoding signatures of selective attention

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 04.03.2015

Thesis supervisor and advisor:

Prof. Lauri Parkkonen

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Mainak Jas

Title: Real-time machine learning of MEG: Decoding signatures of selective attention

Date: 04.03.2015 Language: English Number of pages: 9+42

Department of Information and Computer Science

Professorship: Biomedical Engineering Code: F3001

Supervisor and advisor: Prof. Lauri Parkkonen

Brain–computer interfaces (BCIs) provide disabled patients with access to communication tools and control of prosthetic devices. Most BCIs employ a machine-learning algorithm which analyzes brain data in real time and provides users with feedback.

Magnetoencephalography (MEG) is a non-invasive method which records neuromagnetic signals from the brain at a high temporal resolution. This makes it particularly suitable for real-time analysis and machine learning. Developing tools that allow such analysis will have long-term benefits in using MEG for BCI approaches and exploring new experimental paradigms.

In this thesis, a real-time analysis pipeline for machine learning in MEG was developed with the goal to enable BCI in MEG systems. The implementation details of the pipeline were described in the thesis along with performance details. Additionally, pilot measurements to decode auditory attention were conducted. The spatio-temporal dynamics of the offline experiment were used to optimize the preprocessing steps required for the BCI application. In particular, the frequency range of 1.0–1.5 Hz was found to be particularly discriminative. Finally, simulating this pipeline in pseudo real-time mode demonstrated that a BCI to decode auditory attention is feasible in MEG.

Keywords: machine learning, selective attention, brain–computer interface, magnetoencephalography, real-time analysis

Preface

First and foremost, I want to thank my advisor Professor Lauri Parkkonen. Lauri entrusted me with a challenging project and served as a guiding beacon for over two years. Decoding even the most complex engineering systems and algorithms felt like a breeze with Lauri's support around the corner. I thank Academy Professor Riitta Hari who first invited me to Helsinki as a research intern over three years ago. Her presence in the lab has been truly inspiring. My sincere thanks to Professor Devi Parikh for offering me an exciting spring internship in her group. I thank Professor Alexandre Gramfort, Denis Engemann, Eric Larson, Martin Luessi and rest of the MNE-Python team for two exciting Google Summer of Code internships and a coding sprint in Paris.

I cannot thank enough my mentor, colleague and friend Pavan Ramkumar for igniting my passion in scientific research. I thank my office mates – Eeva, Siina and Mia, my colleagues in the lab – Mathieu for being a fantastic table tennis rival, Harri for many funny conversations and others for lunch conversations. Also, I must thank Ivan for his many interesting machine-learning questions and logical puzzles. I thank my project partners Anne and Dovilė for their support and patience. I thank the laboratory IT support staff and engineers – Petteri, Miro and Ronny for helping me whenever computers crashed or cables were missing. My thanks to Mia for help in conducting my experiments.

I thank my Otaniemi Indian friends – Eldrich, Udit, Palash, Bastian, and Devi to name a few, for mouth-watering Indian food and making me feel at home even thousands of miles from India. My thanks to my international friends – Vertti for introducing me to everything Finnish and Bahram for many philosophical discussions. I also thank Cathy for helping me a lot with my presentations, language corrections, interesting projects, brainstorming ideas and of course, her puns. Each one of these individuals and many others made my thesis writing experience immensely enjoyable.

Last, but not the least, my thanks to my family – my parents for always standing by me, and my sister for her lively demeanour and her sense of humour.

Otaniemi, 04.03.2015

Mainak Jas

Contents

List of Figures

List of Tables

List of Code Snippets

Symbols and abbreviations

Abbreviations

API	Application programming interface
BCI	Brain–computer interface
BSD	Berkeley software development
DAT	FieldTrip data matrix
dSPM	dynamic statistical parameter mapping
EEG	Electroencephalography
EOG	Electrooculogram
EVT	FieldTrip event structure
fMRI	functional magnetic resonance imaging
HDR	FieldTrip header structure
HPI	Head position indicator
ICA	Independent component analysis
IP	Internet protocol
MEG	Magnetoencephalography
MMN	Mismatch negativity
MNE	Minimum norm estimate
PET	Positron emission tomography
SPECT	Single proton emission tomography
SQUID	Superconducting quantum interference device
SSS	Signal space separation
PCA	Principal component analysis
SVM	Support vector machine
TCP	Transmission control protocol

1 Introduction

1.1 Motivation

Magnetoencephalography (MEG) is a versatile neuroimaging tool which offers a high temporal and a reasonably good spatial resolution. Despite the fact that the high temporal resolution of MEG makes it particularly suitable for real-time analysis, no significant progress has been made towards developing tools and techniques that enable such analysis.

The bulk of real-time neuroimaging research in the past few decades has been focussed around electroencephalography (EEG), which measures the electrical signals from the brain on the scalp, primarily because it is portable and relatively inexpensive thus suitable for long-term use. Unfortunately, EEG signals have the disadvantage that they are smeared due to the intervening skull leading to a low spatial resolution. This has hindered development of brain–computer interface (BCI) and related real-time methods which require a high spatial resolution. Despite this, one of the most successful BCI methods (Common Spatial Pattern) in EEG computes a projection that maximizes the differences in the voltage topographies between two conditions. Other neuroimaging modalities with a high spatial resolution suffer from the problem of low temporal resolution, making them unsuitable for real-time analysis.

Therefore, the need of the hour is to develop real-time tools in MEG. Combining machine-learning with real-time analysis will pave the way for online learning in MEG which will have applications in BCI and novel experimental paradigms. This can benefit patients suffering from stroke, disorders of consciousness, amyotrophic lateral sclerosis *etc.* MEG has the added benefit of measuring similar neural sources as EEG. This would make it possible to train models in MEG and use them in EEG for prediction after transforming the models suitably.

1.2 Contributions of the thesis

The main contribution of the thesis is the development of a real-time pipeline for machine learning in MEG data. The work builds on existing software: FieldTrip, MNE-Python and the scikit-learn machine learning package. A real-time and machine-learning analysis pipeline is developed as part of the thesis, whose application programming interface (API) is described in the thesis. Furthermore, a real-time feedback mechanism is also exposed. Finally, attention decoding is presented as a case study of application to real MEG studies. Signatures of decoding attention in MEG – the frequency, channels and time points necessary for optimal classification, are uncovered.

2 Background

2.1 The human brain

The human brain is the most complex part in the human body with 10^{12} neurons in the Central Nervous System, and 10^{15} synaptic connections absorbing and releasing 10^{18} neurotransmitters and neuromodulators per second. The average human brain weighs only 1.5 kg and consumes only 15–20 W of power (Faugeras et al., 1999).

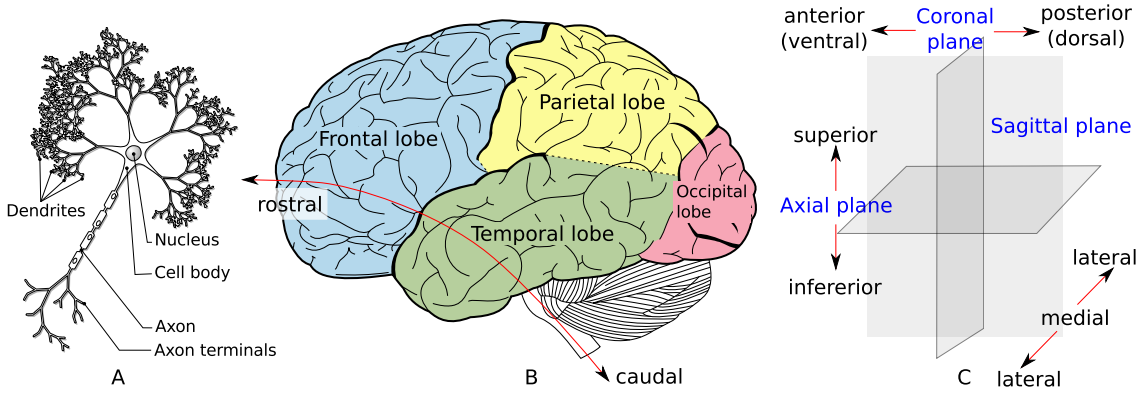

Figure 1: The human brain. A. The structure of a neuron. B. The organization of the cerebrum. C. The coordinate system and naming conventions.

The cerebral hemispheres (the cerebrum; Figure 1B), located above other brain structures, form the largest part of the brain. They are covered by a cortical layer (the cerebral cortex) which is 2–4 mm thick. This layer is folded into *gyri* (ridges) and *sulci* (furrows) to pack maximum amount of surface area of the cortex in a limited volume. The evolutionarily older structures lie below the cerebrum and play an important role in regulating vital bodily functions. The brainstem lies below the cerebrum and is structurally contiguous with the spinal cord.

Cerebrum: The cerebrum consists of the left and right hemispheres which are further divided into four lobes: the frontal, parietal, temporal, and occipital. The left hemisphere is functionally dominant for language and speech whereas parts of the right hemisphere are dominant in performing tasks related to spatial processing and music. Each of these lobes is associated with distinct types of information processing. The frontal lobe is responsible for reasoning, planning, movement, emotions and problem solving. The parietal lobe is associated with movement and orientation. Processing of auditory stimuli, memory and speech are primary functions of the temporal lobe. Finally, the occipital lobe is responsible for visual processing.

Neurons: As with other parts of the body, the brain is composed of different types of cells. The majority of these cells are glia which support the structure and metabolic functioning of the brain. However, neurons, which are much less abundant

compared to the glia cells, are considered the basic building blocks of brain function as they are critical for processing and transmitting information in the brain. The main parts of a neuron are the dendrites, cell body (or soma) and axon (Figure 1A). The dendrites receive inputs from other neurons through *synapses* which are the points of contact between dendrites and axon terminals. The neuron that receives the signal is called the postsynaptic neuron and the neuron which sends the signal is called the presynaptic neuron. At chemical synapses, neurotransmitters are emitted by the presynaptic neuron to transmit the signal. The dendrites are connected to the soma of the cell. It contains the nucleus and is responsible for metabolism of the cell. The signal is transmitted from the dendrites to the cell body. A neuron may receive signals from other neurons and these are accumulated until a threshold is reached. Once the threshold is exceeded, the neuron generates a signal (known as *action potential*) which is transmitted to other neurons through the axon.

Coordinate system: In subsequent sections, it may be necessary to refer to different locations in the brain. Therefore, it is essential to grasp the various coordinate systems in use in neuroscience literature. The most commonly used coordinate system is defined by three planes (Figure 1C) – the axial plane which is parallel to the ground and separates the top (superior) from the bottom (inferior), the coronal plane which is perpendicular to the ground and separates the front (anterior) from the back (posterior), the sagittal plane which separates the left from the right. To define positions closer to the midline along the sagittal plane, the terms lateral (closer to the surface/ear) and medial (closer to the midline) are used.

In addition to these terms, some nomenclature is borrowed from animal literature. For example, rostral refers to structures closer to the forehead whereas caudal refers to structures closer to the tail (Figure 1B). As a result, this axis is curved. Similarly, brain areas can be defined as ventral if they are closer to the abdomen and dorsal if they are closer to the back. Figure 1C summarizes the different naming conventions for locating structures in the brain.

2.2 Consciousness and its disorders

As described in Introduction (Section 1), one motivation behind developing real-time analysis tools in MEG is to create clinically viable solutions for patients suffering from disorders of consciousness. In this context, consciousness refers to the state of being aware of the surroundings or of oneself. Various theories have been proposed to explain consciousness. According to Baars' Global Workspace Theory (Baars, 1993), it is a phenomenon that emerges due to a pattern of simultaneous activity throughout the brain. Another idea that has gained traction is that consciousness is supported by high-frequency oscillations in brain activity. Regardless of the origins of consciousness, we are interested in medical disorders of consciousness which are characterized by the amount of consciousness present, and developing communication channels for patients who are conscious but unable to perform any motor acts.

Medical conditions that inhibit consciousness are considered disorders of consciousness (Naci et al., 2012). They can arise from either a progressive brain disorder

or acute brain injury. Progressive brain disorder may lead to advanced amyotrophic lateral sclerosis and consciousness is considered to be present in these patients. However, patients who have suffered an acute brain injury may fall into several categories depending on their level of responsiveness: acute locked-in syndrome, minimally conscious, vegetative state to chronic coma in rare cases. Chronic coma is characterized by lack of awareness and no spontaneous eye opening. In the vegetative state, patients demonstrate stimulus-induced eye opening but no awareness of their environment. Minimally conscious patients exhibit small and inconsistent signs of awareness. Finally, those in the locked-in state demonstrate consistent signs of awareness through small but reproducible movements.

2.3 Attention and dichotic listening

A closely-related phenomenon to consciousness is attention. Selective attention refers to the ability of the brain to filter out irrelevant information and only consciously perceive the relevant signals from the external world. This is a very important trait as it allows the brain to focus in the presence of many salient distractors. Koch and Tsuchiya (2007) have argued that attention and consciousness are distinct phenomenon. They laid down experimental conditions where attention to an external stimuli can be present with or without the subject being consciously aware of it. Thus, by careful selection of experimental conditions, these two effects can be dissociated from each other.

One such paradigm is dichotic listening. The classic dichotic listening studies (Cherry, 1953) were motivated by the cocktail party effect. Cherry (1953) made the interesting observation that a person in a cocktail party can easily focus his/her attention on the person speaking to him/her while tuning out sounds from all the other people present in the party. This led him to conduct an experiment where subjects were simultaneously presented with two auditory stories, one on each ear. The subject was asked to follow or shadow (repeat) the story word by word or phrase by phrase. As a consequence of these experiments, he outlined five major cues that play a role in auditory attention: i) the direction from which the sounds originate, ii) lip reading and gestures, iii) differences in voices (*e.g.* mean pitch, mean speed, male and female *etc.*), iv) differences in accents, and v) transition probabilities (voice dynamics, subject matter *etc.*). The experiments suggested that the physical characteristics of the unattended stream were perceived but the semantics were not recognized.

This observation has inspired many neuroimaging experiments to use the cocktail party effect in dichotic listening studies to investigate attention(see *e.g.* Hill and Miller, 2009; Maddox et al., 2012; Zion Golumbic et al., 2013). In particular non-invasive measurements such as MEG (see Section 2.6 for an overview) can capture the oscillatory dynamics of attention. For example, there are several studies(*e.g.* Jensen et al., 2012; Bonnefond and Jensen, 2012; Wilsch et al., 2014) which show that alpha and gamma oscillations are critical to maintaining attention and blocking out distractors. Finally, MEG, while offering an acoustically quiet environment when compared to other non-invasive techniques such as fMRI, also has a higher spatial

resolution compared to EEG.

2.4 Oddball paradigms

The brain is known to respond to novel sounds using two distinct dynamic modes (King et al., 2014): the mismatch negativity (MMN) and P300. The MMN, first discovered by Näätänen et al. (1978), is an electrophysiological signal that is generated over the superior temporal areas whereas the P300 is distributed over the frontal, parietal and temporal lobes. While the MMN does not depend on the subject's state of consciousness, the P300 requires the subject to consciously detect the novel stimuli. These two EEG components are thought to relate to different computations that the brain performs. While the MMN reflects a prediction error, *i.e.* sensory input differs from the internally generated prediction, the P300 is related to a working-memory update which produces a longer and sustained signal. The MMN is thought to be a result of a fast serial process peaking around 200 ms after the onset of a deviant stimulus where the prediction errors propagate through a series of areas until the internal model cancels out the prediction error, whereas the P300 is a stable and slow activation reaching its maximum at around 300 ms. This suggests that a paradigm based on P300 will be successful in detecting awareness in locked-in patients.

Interestingly, the amplitude of P300 is affected by the sequence of stimuli (Squires et al., 1976). When a sequence of low- and high-pitched stimuli were presented to subjects, the response was found to depend on three factors: the memory for the particular event within the preceding sequence, the structure of the preceding sequence and the global probability of the event. Furthermore, a regression analysis performed on these factors was able to explain a high percentage of the variance. This study has inspired more advanced models (Mars et al., 2008; Kolossa et al., 2012) based on temporal filters to explain the P300 amplitude fluctuations.

2.5 Brain–computer interfaces

Since the 1980s, Brain–computer interfaces (BCIs) based on electroencephalography (EEG) have enabled clinical patients to communicate and control external devices. Established BCI methods exploit known properties of electrophysiological signals: slow cortical potentials, P300 potentials, and mu or beta rhythms.

Despite tremendous progress in BCI techniques over the last decades (Wolpaw et al., 2002), most of these works do not address communication for severely disabled patients such as locked-in patients (see Section 2.2). Often, these patients lack the ability to track or even fixate on objects with their eyes. This makes it difficult to successfully deploy traditional BCI paradigms with such patients. The first evidence of awareness in locked-in patients was found in a functional magnetic resonance imaging (fMRI) study by Owen et al. (2006) where patients were asked to imagine hitting a tennis ball to answer "yes" or walking from room to room in their house to answer "no". The two mental tasks activated different areas of the brain depending on the desired response. A consistent response pattern in about 20–25% of these patients indicated that they were still conscious even though there were no externally

visible signs indicating awareness. However, this paradigm cannot be taken into clinical practice. This study spurred efforts towards bedside detection of awareness with EEG (Cruse et al., 2011) to develop a communication interface but this has been mired in controversy (Goldfine et al., 2013).

Therefore, the evidence points towards moving away from fMRI into other non-invasive methods using non-traditional approaches. For example, the focus has shifted from attending a specific type of stimuli in an oddball experiment to attention in a crowded cocktail party scene (Schreuder et al., 2010; Nambu et al., 2013; Gao et al., 2011; Matsumoto et al., 2013). In such an approach, the auditory stimuli are delivered from multiple sources and the subject is asked to attend to one of them. This has proved to be very effective and in the study by Nambu et al. (2013), as many as 6 directions could be decoded with a classification accuracy of around 70%. We can note that in MEG, this approach poses an additional challenge because auditory stimuli can be delivered only through earphones. Therefore, the cocktail party effect must either be synthetically produced or reproduced from recordings of dummy head models (as described in Section 3.4). Finally, this approach has the advantage that the bit-rate (which quantifies the throughput of information transfer) can be easily scaled, within limits, by increasing the number of sound sources in the cocktail party scene.

2.6 Magnetoencephalography

We have described previously (in Sections 2.3 and 2.5) that using MEG for our experiment has several advantages over other non-invasive measurement techniques. We will now explain the theory, instrumentation and standard methodology of analyzing MEG data.

MEG is a technique for investigating weak magnetic fields originating from neural currents in the human brain. Sensors based on superconducting quantum interference devices (SQUIDs) placed close to the brain pick up these magnetic fields. Synchronous activity of tens of thousands of pyramidal cells (neurons with a pyramid-shaped soma and two distinct dendritic trees) are required in order to generate fields strong enough to be measured by these sensors.

Most neuroimaging modalities are graded according to their temporal and spatial resolution. The rate at which brain activity can be measured determines the temporal resolution while the accuracy with which this activity can be localized determines the spatial resolution of the modality. Figure 2 shows a comparison of various neuroimaging methods in terms of their spatial and temporal resolution. MEG as well as EEG can measure activity in the brain with a temporal resolution of the order of 1 ms. This is unlike positron emission tomography (PET), single photon emission tomography (SPECT) or fMRI. As fMRI measures changes in the blood flow and oxygenation levels, the temporal resolution of this method is limited to about 1 s. The temporal resolution of PET is of the order of 0.1 s but limitations are imposed due to the radiation dosage that can be administered on the participant. It can be noted that this diagram is only a schematic and not drawn to scale.

The first MEG measurement (Cohen, 1968) used a single induction coil to detect

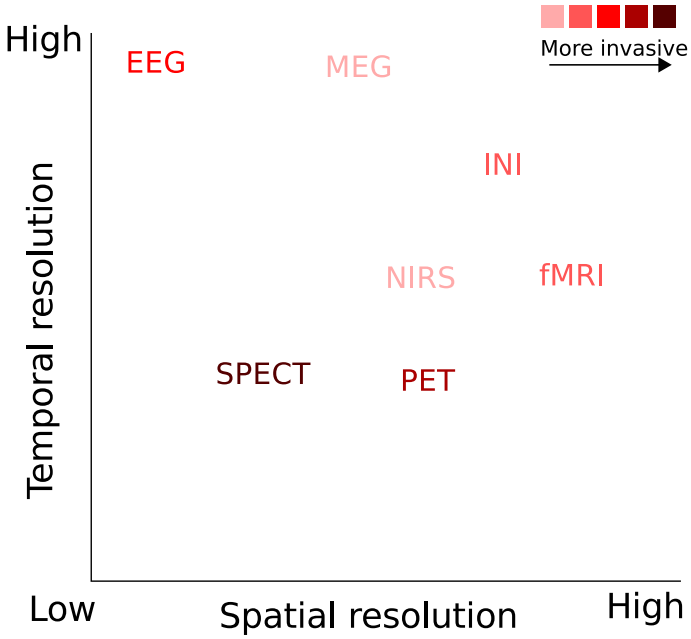

Figure 2: Neuroimaging methods differ in terms of the information they measure. MEG has a high temporal resolution and a medium-to-high spatial resolution (adapted from Jääskeläinen (2012)). MEG=magnetoencephalography, EEG=electroencephalography, NIRS=near-infrared spectroscopy, PET=positron emission tomography, SPECT=single photon emission tomography, and INI=inverse imaging, a method to speed up acquisition of fMRI images.

alpha rhythm in the brain. These measurements were very noisy and needed thousands of averages to filter the signal from the noise. This is because MEG is sensitive to various kinds of interfering magnetic fields which are orders of magnitude larger than the biomagnetic fields. The earth's magnetic field, electric disturbance, traffic etc. are some examples of noise sources. In addition, physiological artifacts such as eyeblinks, heart beats, muscle artifacts, head movements etc. also corrupt the signal adversely. Therefore, modern MEG devices employ a careful mix of hardware instrumentation and software tools to effectively suppress noise while being sensitive to the brain's magnetic field.

2.6.1 Instrumentation

Figure 3 shows a schematic of an MEG acquisition device. The SQUID sensors are submerged in liquid Helium to maintain superconductivity. The sensor array consisting of around 300 sensors is housed in a helmet shaped dewar, covering most of the head. The MEG measurements are usually performed in a magnetically shielded room constructed from layers of μ-metal and aluminium. The μ-metal is effective in suppressing low frequencies of the external interference while the aluminium is more effective for the high-frequency band above about 10 Hz. This is known as "passive" shielding. In yet another design of shielded rooms, the external magnetic field is

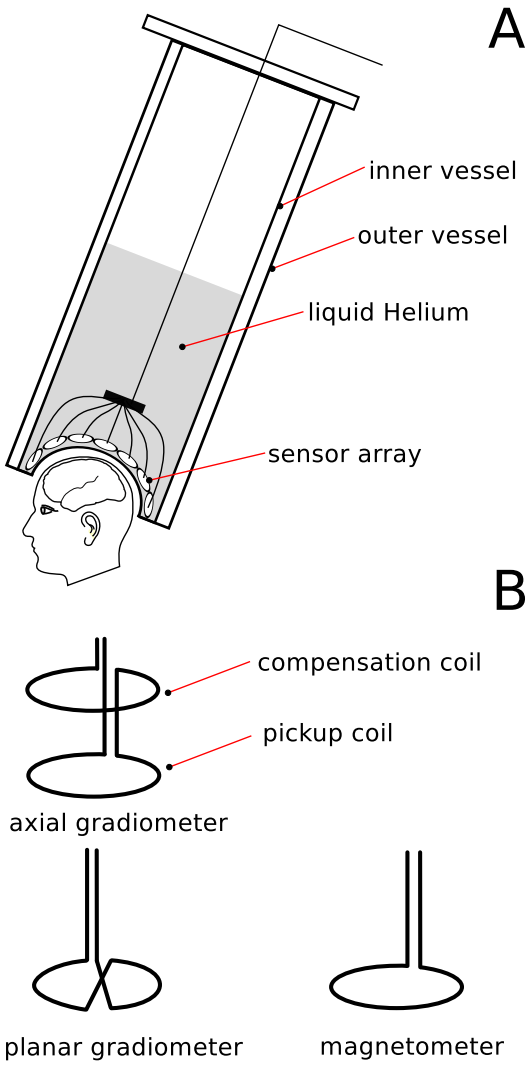

Figure 3: A. Schematic of an MEG acquisition device (adapted from Hansen et al. (2010)). The SQUID sensors are submerged in liquid Helium to maintain superconductivity. Modern MEG sensor arrays contain upto 102 magnetometers (to measure magnetic field) and 204 gradiometers (to detect spatial derivative of magnetic field) B. Typical magnetometer and gradiometers used in the sensor array.

detected and currents are sent around the room to generate a net magnetic field which cancels the external field. This method is known as "active" shielding and it provides better noise cancellation when used in combination with passive shielding.

Sensors: Modern MEG devices contain magnetometers, gradiometers, or a combination of them. Magnetometers measure a component of the magnetic field whereas the gradiometers measure the spatial derivative of the field. Figure 3 shows that the first order axial gradiometer consists of a pickup coil and a compensation coil. These two coils have the same area and are connected in series but wound in opposite

directions. Therefore, spatially uniform background will be cancelled out by the compensation coil. However, closer magnetic sources will produce a much higher field in the pickup coil compared to the compensation coil, thus producing a net signal. This results in reduced sensitivity for distant sources in the gradiometer. Therefore, the gradiometer is particularly suitable to detect superficial sources in the brain, but a magnetometer can be used for deeper sources as well.

2.6.2 Noise suppression

In addition to these hardware-based methods, clever signal processing techniques can be used to retain the interesting information while throwing out the noise. The Signal Space Separation (SSS) method (Taulu et al., 2004) and its temporal extension (tSSS) can do exactly this by suppressing external interference, standardizing the MEG signal with respect to the head position and sensor geometry, and compensating for head movements. The main idea behind this method is that the field measured at the sensors can be expressed as two multipole series expansions of spherical harmonic basis functions. The two expansions are for magnetic sources inside the sensor helmet and for sources outside the helmet. Since these subspaces are linearly separable, the contributions from magnetic sources outside the sensor helmet can be removed by simply discarding these components. Furthermore, the spatial sampling theorem imposes a theoretical limit on the number of components (around 100) that are necessary to maintain a compact representation of the brain signal.

The SSS method is also useful for head movement compensation during MEG recordings. Continuous tracking of the head movement is done using Head Position Indicator (HPI) coils (Ahlfors and Ilmoniemi, 1989) which are attached on the scalp and excited at typically around 290–330 Hz. Localizing these coils and thereby the head position enables transforming the MEG signals to a reference head position. This is done by expressing the signal as a multipole expansion tied to the origin of the current head position, and then reconstructing the signal at the sensors for the reference head from the multipole components.

2.6.3 Filtering and artifact removal

Bad sensor channels can be identified by manually inspecting the MEG signals. Multiple strategies are useful for artifact removal. The simplest of these is to discard segments of the signal in individual channels if the peak-to-peak amplitude exceeds a certain threshold. Artifacts due to eyeblinks and saccades can be similarly removed by analyzing the peak-to-peak amplitude in the horizontal and vertical electrooculogram (EOG) channels. If the EOG channel was not available during the measurement, ocular artefacts can still be suppressed by decomposing the signal using principal component analysis (PCA) or independent component analysis (ICA) and by discarding the component that correlates best with a channel showing the artefact clearly (usually a frontal channel).

Most of the biomagnetic signals are bandlimited to the lower end of the spectrum. At very low frequencies ($<$1 Hz), noise due to the environment and physiological sources such as cardiac muscle can contaminate the signal. At very high frequencies,

the instrument noise plays a major role. Filtering the signal in each sensor temporally can reduce this noise.

2.7 Machine learning

Machine-learning approaches, also known as multivariate pattern analysis (MVPA) are becoming increasingly popular in neuroscience. This is because MVPA methods are more sensitive compared to traditional univariate methods of analysis – they can pick up patterns in variations across multiple signals which a univariate method cannot. The MVPA analysis can be based on either encoding or decoding models (Naselaris et al., 2011). Decoding models attempt to predict the experimental stimuli given the brain activity whereas encoding models try to describe the activity at each voxel in the brain as a function of the stimuli. Although these two approaches are complementary, Naselaris et al. (2011) have argued that encoding models are more powerful as they provide a method to directly refer to the underlying brain activity. Also, deriving a decoding model given the encoding model is much more straightforward than the other way round.

One of the first decoding studies that gained limelight was by Haxby et al. (2001) which showed that the patterns of responses in fMRI recordings could be used to discriminate between faces, cats, man made objects and nonsense pictures. The distinction between these categories was not simply because each category was represented in a different region of the brain. In fact, even when the region to which each category responded maximally was excluded from the analysis, the stimulus class could be identified from the brain response. Cox and Savoy (2003) performed a similar experiment where they decoded object categories on a trial-by-trial basis. These two studies showed that neuroimaging methods contained far more information than earlier thought and this opened up the field for many more MVPA studies.

In an interesting study along similar lines, Kamitani and Tong (2005) classified stimulus orientation from recordings in the primary visual cortex. Furthermore, they were able to detect feature-based attention to one of the orientations using their classifiers. In another groundbreaking study, Kay et al. (2008) developed an encoding model based on receptive-field models and used that to identify natural images from a large dataset. In a similar vein, Mitchell et al. (2008) were able to predict fMRI activation for nouns with a model trained from a large text corpus.

Wessberg et al. (2000) have demonstrated that such techniques can be used to great effect in BCI applications, even those involving very complex movements, such as reaching out for objects in three-dimensional space. In their experiments with monkeys, they developed a procedure for real-time control of a robotic arm. Both a linear and an artificial neural network (ANN) model were used to predict the one- and three-dimensional trajectory of hand movement from cortical signals.

Information-based brain mapping (Kriegeskorte et al., 2006; Ramkumar et al., 2013) also make use of multivariate analysis methods. Instead of averaging the signal of interest, a temporal or spatial "searchlight" is used to scan and localize informative parts of the data.

Finally, machine learning has also found applications as an indicator of the stage

of a disease. Tzovara et al. (2013) modelled the scalp voltage topographies in EEG using a mixture of Gaussians and used it to predict discrimination between oddball and standard stimuli. The prediction accuracy was indicative of the stage in which patients were during acute coma. Machine learning has also been applied to predict the onset of epilepsy (Golestani and Gras, 2014).

2.7.1 Supervised learning

Learning models can be broadly classified into two types – *supervised* and *unsupervised*. In unsupervised learning, the goal is to discover structure in unlabelled data. A familiar example of this type of learning is clustering. Applications such as online news services, which need to organize a large number of news items automatically according to some underlying hierarchy, can benefit form unsupervised learning. On the other hand, supervised learning is useful when a mapping from data $\boldsymbol{D}$ to labels $\boldsymbol{y}$ needs to be learnt. A commonly cited example of supervised learning is predicting the species of an iris flower based on measurements of the sepal and petal lengths. This thesis will mainly focus on supervised learning.

Training and test data: It is considered common practice to not test the performance of a supervised learning algorithm on the same data it was trained on. The reason is that this would not be evaluating the performance on new data. Therefore, the entire dataset is split into a training set and test set. The test set is available for prediction only after the model has been learnt.

Overfitting: The need for a separate training and test set can be understood by considering the examples in Figure 4. The data can fit very well to a polynomial function of degree $d = 5$, which may lead to the erroneous conclusion that this function is representative of the underlying data. Attempting to predict new labels based on this function will lead to a high error rate. Therefore, one must ensure that the model *generalizes* well.

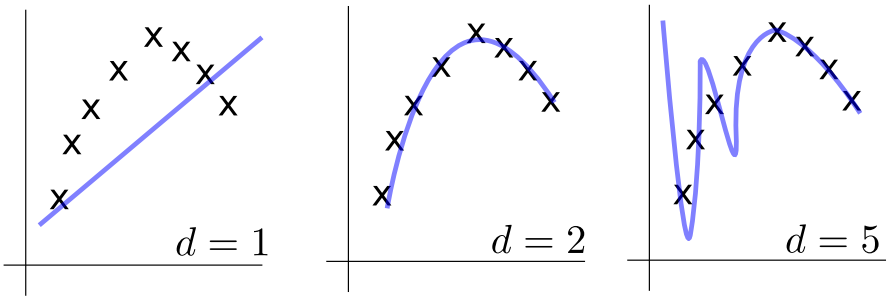

Figure 4: Toy example of overfitting. A polynomial of degree $d = 1$ does not fit the data very well but polynomials of degree $d = 2$ and $d = 5$ can both explain the data. The higher order polynomial will not generalize well.

Cross validation: One pitfall of splitting the data once into a training and test set is that the choice of the split may affect the performance. To ensure that the model generalizes well, it is important that the performance measure is robust. One approach is to split the training data further into a training set and test set (known as the *validation* set). The performance is tested on the validation set, repeated multiple times and the results are averaged. Depending on how the splits are done, this can lead to different types of cross-validation procedures. The most common ones are k-fold cross-validation and leave-one-out cross-validation (LOOCV). In k-fold cross-validation, the data are divided into k parts. One of these parts is used for validation and the other $k-1$ parts are used for training. This is repeated k times until all the parts are exhausted and used for validation. LOOCV is a special case of k-fold cross-validation where k is equal to the number of data samples and each of the parts is a sample in the dataset.

Hyperparameter optimization: Often, a model which has unknown parameters must be tuned. For example, in Figure 4, the degree of the polynomial which must be fit to the data is unknown. To overcome this problem, cross-validation can be performed on the training data with all viable parameters and the model which performs the best is used for testing. When there is more than one parameter, all possible combinations of values for the parameters are used to fit models and their performance is evaluated. Such an exhaustive search procedure is also known as *grid search*.

2.7.2 Support vector machine

One of the most common supervised learning algorithms is Support Vector Machine (SVM) which was first proposed by Cortes and Vapnik (1995). The principle of the algorithm is to find an optimal hyperplane which maximally separates the two classes. This is achieved by identifying support vectors which lie on the boundary of each class and finding a linear hyperplane which produces the maximal margin. Figure 5 graphically illustrates this idea using a toy example. Non-linear classification can be done by mapping the data samples into a higher dimensional space where the data samples are linearly separable. This is known as the *kernel trick*. SVMs can be used for both classification and regression.

In the most basic mathematical formulation of the problem, the training data $\boldsymbol{D}$ is a set of n points belonging to a p-dimensional feature space. Each sample $\boldsymbol{x_i}$ in the training set is accompanied by a label y_i. Any hyperplane passing through the set of points $\boldsymbol{x}$ will satisfy

$$\boldsymbol{w}\cdot\boldsymbol{x} - b = 0, \tag{1}$$

where $\cdot$ denotes the dot product and $\boldsymbol{w}$ the normal vector to the hyperplane. If the data samples are assumed to be linearly separable and the labels can take on values 1 and -1, the hyperplanes lying on the margins (as shown in Figure 5) can be described using the equation

$$\boldsymbol{w}\cdot\boldsymbol{x} - b = \pm 1. \tag{2}$$

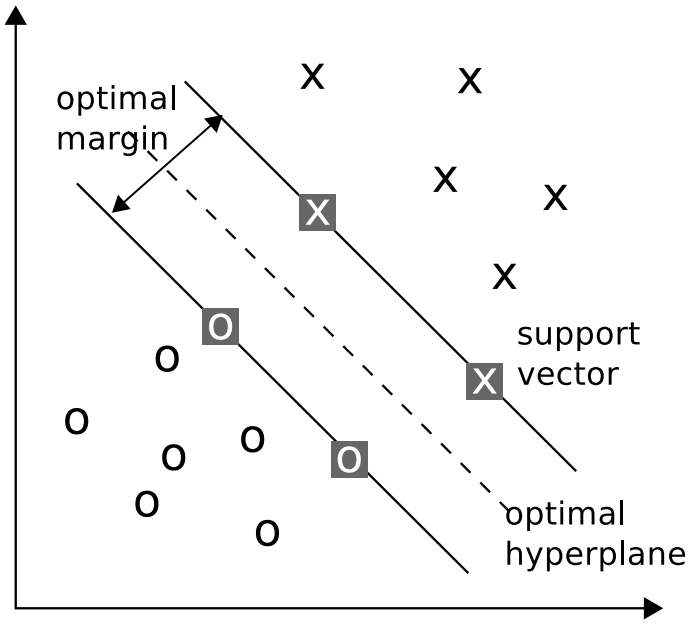

Figure 5: A toy example in two dimensions to show how SVM classifies data. The data points of the two classes are shown as crosses and noughts.

Since the unit vector perpendicular to these hyperplanes is described by $\frac{1}{\|\boldsymbol{w}\|}$, the distance between the hyperplanes is $\frac{2}{\|\boldsymbol{w}\|}$. To prevent data points from falling into the margin, they must satisfy the constraints

$$\boldsymbol{w}\cdot\boldsymbol{x_i} - b \geq 1 \tag{3}$$

for $\boldsymbol{x_i}$ belonging to the first class, and

$$\boldsymbol{w}\cdot\boldsymbol{x_i} - b \leq -1 \tag{4}$$

for $\boldsymbol{x_i}$ belonging to the second class. Equations 3 and 4 can be combined in a single equations as

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x_i} - b) \geq 1, \text{ for all } 1 \leq i \leq n. \tag{5}$$

Therefore, this boils down to an optimization problem to minimize $\|\boldsymbol{w}\|$ subject to the constraint posed by Equation 5. By substituting $\|\boldsymbol{w}\|$ for $\frac{1}{2}\|\boldsymbol{w}\|^2$ and using Lagrange multipliers $\boldsymbol{\alpha}$, this can therefore be expressed as

$$\arg\min_{\boldsymbol{w},b}\max_{\boldsymbol{\alpha}\geq 0}\left\{\frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i[y_i(\boldsymbol{w}\cdot\boldsymbol{x_i} - b) - 1]\right\}. \tag{6}$$

This is a quadratic optimization problem which can be solved using standard techniques. The weights of the SVM can be shown to take on the form

$$\boldsymbol{w} = \sum_{i=1}^{n}\boldsymbol{\alpha}_i y_i \boldsymbol{x}_i, \tag{7}$$

where $\boldsymbol{x}_i$ are the support vectors.

3 Materials and methods

3.1 Software packages

Here, various software packages that are useful for real-time machine learning applications in MEG are described. The major contributions of this thesis are in the `realtime` and `decoding` modules in the MNE-Python software package.

The `realtime` module in MNE-Python communicates with the the FieldTrip buffer (Section 3.1.1) to obtain the data in real time. Next, data preprocessing and feature extraction can be performed using the advanced functions available in mne-python. Finally, these features can be forwarded to the scikit-learn software package through the `decoding` module, completing the pipeline for real-time machine learning. First, each of these packages will be introduced – FieldTrip buffer, mne-python and scikit-learn – and then the additions to the MNE-Python package will be explored in more detail.

3.1.1 FieldTrip buffer

Request	Purpose
PUT_HDR	Put header information in the buffer
GET_HDR	Get header information in the buffer
FLUSH_HDR	Flush header information in the buffer
PUT_DAT	Put data samples in the buffer
GET_DAT	Get data samples in the buffer
FLUSH_DAT	Flush data samples in the buffer
PUT_EVT	Put events in the buffer
GET_EVT	Get events in the buffer
PUT_EVT	Flush events in the buffer
WAIT_DAT	Wait for samples and/or events

Table 1: FieldTrip buffer protocol.

The FieldTrip real-time buffer is a generic and vendor-independent interface towards real-time analysis. The buffer is part of the FieldTrip software package (Oostenveld et al., 2010) which is used for analysis of MEG, EEG and invasive electrophysiological data. rtMEG (Sudre et al., 2011) is a proxy program that links a 306-channel Elekta Neuromag MEG device to the FieldTrip buffer. The delay introduced by this program is less than 50 ms which is sufficiently low for most applications. The proxy program also implements the FieldTrip buffer and includes C++ functions for reading the buffer. Python bindings are already available and included as part of the program.

The FieldTrip buffer contains three key elements: the *header structure* (HDR), the *data matrix* (DAT) and the *events structure* (EVT). The data matrix is implemented

as a ring buffer which means that it has only a finite capacity to store old samples. For communication, the client (which receives the data from the buffer) always sends a request to the server (the buffer) and the server responds to the request. A request, for example, could be GET_DAT and the server will respond with the data matrix. Table 1 presents a comprehensive list of requests that can be issued by the client.

3.1.2 MNE-Python

The MNE software package (Gramfort et al., 2013) is a set of tools for carrying out advanced MEG and EEG analysis. The software package derives its name from the namesake analysis technique, Minimum Norm Estimate (MNE) which was proposed by Hämäläinen and Ilmoniemi (1994) to compute estimates of neural currents from MEG measurements. MNE software includes methods for preprocessing, source estimation, time–frequency analysis, statistical analysis, and functional connectivity. The MNE software consists of three software packages which are shipped together: MNE-C, MNE-Matlab and MNE-Python. The MNE-Python package is a collaborative open source effort with contributors from around the globe. The entire source code is available publicly on github[1] and licensed through a very permissive BSD license.

The main container objects in MNE-Python are `Raw`, `Epochs` and `Evoked`. The `Raw` object is used to manage unfiltered data read in from a `fiff` file. All the containers share an `info` attribute which stores the measurement meta data as a python dictionary. This includes the measurement date, subject and experimenter names, names of sensor channels and their positions, the head to device coordinate transformation matrices and the passband of the data. The `Raw` object includes methods to filter, crop (retain only the desired time window and delete the rest of the signal), slice (select a subset of channels and time points), and browse the data as well as to reject bad channels interactively *etc*.

The `Epochs` object cuts the data into segments around the triggers of interest and stacks them together into a 3D matrix of shape `n_epochs x n_channels x n_times`. Each of the epochs can be baseline-corrected by subtracting the mean of a window before the stimuli was presented. Rejection parameters can be used to specify the peak-to-peak threshold in the different channels for removing epochs with artifacts such as eye-blinks or muscle movements. If there is more than one experimental condition, each of the conditions is stored as a separate element in a Python list. By taking the mean across the epoch dimension a user can get stimulus-locked averages which have a higher signal-to-noise ratio as compared to a single epoch. This results in an `Evoked` object which has methods for plotting the field lines on the head surface and for visualizing the topographic maps at various time points. Evoked objects can also be subjected to source modelling.

At each stage of the analysis, the sensor channels can be selected by type (magnetometers, gradiometers, EEG *etc*.) or by using a regular expression[2].

[1] https://github.com/mne-tools/MNE-Python

[2] A regular expression is a sequence of characters that form a search pattern

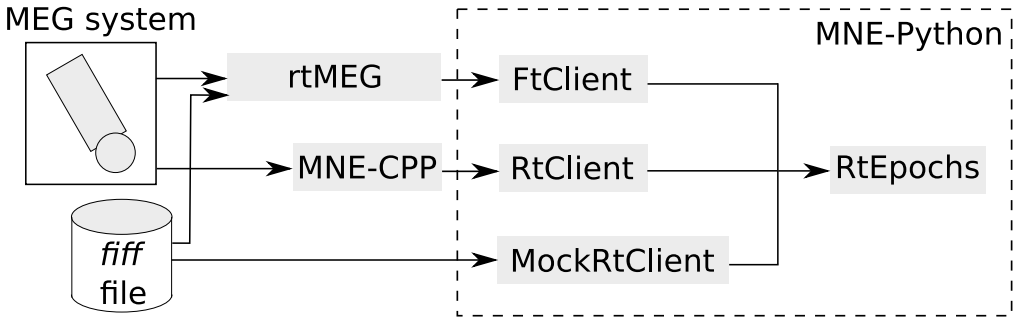

Figure 6: A schematic of the real-time API in MNE-Python.

3.1.3 Scikit-learn

Scikit-learn (Pedregosa et al., 2011) is a python module which implements a wide range of supervised and unsupervised machine-learning algorithms. An `estimator` object incorporates a `fit` and sometimes a `predict` method. The `fit` method accepts a data array and a label array (in the case of supervised learning) estimates the parameters of the model. Some estimators called `transformers` incorporate a `transform` method which can be used for preprocessing the data such as applying principal component analysis (PCA). Both these methods are lumped together by a `fit_transform` which performs both model fitting and transformation.

Scikit-learn estimators can be combined in a `Pipeline` to apply sequential feature extraction steps or in parallel using `FeatureUnion` to concatenate features together. Furthermore, `partial_fit` methods can be used for online learning to partially train a linear model and update it each time a new sample arrives. This makes the package particularly well-suited for real-time machine learning.

3.2 Software structure and interfaces

3.2.1 Realtime client and epoching

The MNE real-time API is designed to handle the entire MEG/EEG/trigger data along with the measurement information in real time. A schematic of the entire system is shown in Figure 6. MNE-Python can receive a real-time data stream in a variety of ways – rtMEG, MNE-CPP and simulated data from a `fiff` file. This data is then sent to the `RtEpochs` object for further processing. The system consists of a real-time server in MNE-CPP and one or more clients which connect to it using Transmission Control Protocol (TCP)/Internet Protocol (IP).

The `RtClient` object establishes the connection to MNE-CPP server so that data flow between the server and the client can be started. The length of requested data segments can be specified while instantiating this object.

From the `RtClient` object, the data are buffered to the `RtEpochs` object which performs epoching on the data for further processing using MNE. The `RtEpochs` object is similar to the `Epochs` object used for offline analysis (see Section 3.1.2).

As in the offline mode, epochs are extracted based on from the trigger channel which fires when stimuli are presented. Options for segmenting the data based on the onset, offset or both onset and offset of the trigger channel are available. It is also possible to define epochs as trigger signal changes between two non-zero values. Events in offline analysis of MNE-Python are a `numpy` array but for real-time analysis, they are implemented as a list of tuples to avoid the overhead due to memory reallocation. The `RtEpochs` and the `Epochs` object inherit from `_BaseEpochs`, therefore making all methods from `_BaseEpochs` available also in `RtEpochs`. When epochs are not received for a certain duration of time (2 s by default), a timeout occurs and the connection is closed. Since an epoch from the `RtEpochs` object is just like an offline `Epochs` object, it can be used with all the other epoch-processing functions in MNE-Python.

It is important that epochs are available as soon as the data are received. If MNE-Python reads the data in segments of L samples and epochs requested are of length N such that $N > L$, a data segment must be maintained internally to ensure that the whole epoch is available. This data segment has a length $k*L$ such that $k*L \geq N+L$.

```
for ii, ev in enumerate(epochs.iter_evoked()):
    if ii == 0:
        grand_ave = ev  # ev is the current epoch
    else:
        grand_ave += ev
```

Code Snippet 1: iter_evoked to accumulate the average during the measurement.

Finally, for computing online averages efficiently, MNE-Python has the capability to maintain a moving average of the epochs. The moving average computes the average of the `numpy` array of shape `n_epochs` x `n_channels` x `n_times` and produces an `Evoked` object which contains a `numpy` array of shape `n_channels` x `n_times`. This is achieved using the iterator `iter_evoked` which converts each epoch to an `Evoked` object with `n_ave = 1`[3]. This can then be conveniently used to compute averages in real time using the arithmetic + operator implemented in the `Evoked` object. Code Snippet 1 illustrates this process.

3.2.2 The FieldTrip connector

Unfortunately, the MNE-CPP real-time server supports only Elekta Neuromag MEG systems. However, the FieldTrip buffer protocol is vendor independent and proxy modules are available for several MEG and EEG systems. Thus, it is thus better suited for carrying out experiments on both EEG and MEG. To take advantage of the FieldTrip real-time buffer, a connector to the FieldTrip real-time client is incorporated in MNE-Python which will guess the measurement information from the FieldTrip header objects. The data are requested using GET_HDR and GET_DAT calls described in the buffer protocol in Section 3.1.1. First, the GET_HDR is called to get the header object including the current sample index. Next, the WAIT_DAT

[3] `n_ave` is the number of epochs used to obtain the average

call can be used to wait till the requested number of samples are available in the buffer. Finally, the GET_DAT call can be used to actually read the data samples.

The API for `FtClient` is identical to the `RtClient`, which means that it can be used together with `RtEpochs`. However, in addition, it offers a `get_data_as_epochs` method which can be used for non-stimulus-locked acquisition of data. This method returns the latest `n_samples` from the buffer as an `Epochs` object, which enables asynchronous, moving-window processing of data with minimal latency. While some samples may be skipped due to computation time, the results are always real-time because only the latest `n_samples` are queried from the buffer using this method.

3.2.3 Decoding module in MNE-Python

Estimator	Purpose
`Scaler`	Standardize the data by mean subtraction and division by standard deviation.
`FilterEstimator`	Apply a zero-phase low-pass, high-pass, band-pass or band-stop filter to the selected channels.
`PSDEstimator`	Estimate the power spectral density (PSD) using multitaper methods (Thomson, 2007).
`ConcatenateChannels`	Concatenates channels into a single vector, thus converting a 3D epoch into a 2D numpy array as a feature vector which is compatible with scikit-learn.
`CSP`	Compute the common spatial pattern (CSP) for MEG data.

Table 2: Description of transformer objects in MNE-Python.

Once the data is available as 3D matrices (epochs, channels, time points), it can be used for machine learning. As discussed in Section 3.1.3, `estimator` objects, which implement a `fit` method, can be inserted in a `Pipeline` to apply feature extraction steps sequentially. The most common feature extraction steps relevant in processing MEG data are filtering and power spectral density estimation, for which estimators were implemented as part of this thesis. `FilterEstimator` and `PSDEstimator` are simply wrappers around the filtering and PSD estimation functions in MNE-Python implementing a `transform` method. Filtering the data after epoching can result in edge artifacts which is an undesirable outcome of using the real-time mode.

Since the magnetometer and gradiometer channels have different amplitudes and noise levels, they must be treated differently during machine learning. The `Scaler` estimator standardizes the data according to different channel types by subtracting the mean and normalizing by the standard deviation of the data in that channel type (magnetometer, gradiometer or EEG). The `ConcatenateChannels` estimator can

be used to convert the 3D data into 2D by concatenating along the channel dimension (`n_channels`). This step is necessary before the data can be used in any of the scikit-learn estimators which accept only 2D matrices of the form `n_samples x n_features`. All the relevant `estimator` classes in MNE-Python are listed in Table 2.

To avoid a hard dependence on `scikit-learn`, the `TransformerMixin` class, from which all `transformer` objects are derived, is copied into the codebase. Wherever possible, *nested imports* are used by importing `scikit-learn` related modules within a function so that only users using those particular functions will need to install `scikit-learn`. *PEP8* conventions[4] are rigorously followed and *pyflake* code checker[5] is used in development.

```
# SVC, ShuffleSplit, Pipeline and cross_val_score
# come from scikit-learn
filt = FilterEstimator(rt_epochs.info, 1, 40)
scaler = preprocessing.StandardScaler()
concatenator = ConcatenateChannels()
svc = SVC(C=1, kernel='linear')

clf = Pipeline([('filter', filt), ('concat', concatenator),
                ('scaler', scaler), ('svm', svc)])
cv = ShuffleSplit(len(y), 5, test_size=0.2, random_state=42)
scores_t = cross_val_score(clf, X, y, cv=cv, n_jobs=1) * 100
```

Code Snippet 2: Combining preprocessing steps into a pipeline.

An example of how the estimators can be combined into a pipeline is shown in Code Snippet 2. Note how the `Pipeline` object can be used in the cross-validation function similar to any `estimator` which has a `fit` and `transform` method.

3.2.4 Real-time feedback

Using the `realtime` and `decoding` modules now present in MNE-Python, one could estimate the performance of a classification pipeline in real-time. However, it is still not possible for the pipeline to adapt according to the performance. For example, if one class is being decoded worse than the other, providing the classifier with more examples of this class can improve the performance. This opens up the possibility of real-time feedback which in the auditory domain can be in the form of changes in pitch, volume or speed of stimulus presentation.

MNE implements two objects to enable real-time feedback – `StimServer` and `StimClient`. The server is multi-threaded with a separate thread handling requests from each client. Communication between threads is achieved using the `Queue` module in Python which implements the First-In-First-Out (FIFO) queue structure. Each client has its own queue, which means that the `StimServer` can handle the connection to each client independent of the others.

[4] https://www.python.org/dev/peps/pep-0008/
[5] https://pypi.python.org/pypi/pyflakes

An `add_trigger` method in the `StimServer` object adds a trigger code (an integer) to the queue of all the clients. The clients can request the oldest trigger code in the queue using a `get_trigger` method. An example of how this can be done is shown in Code Snippet 3.

```
# in the acquisition computer
events = [4, 3, 4, 4, 4, 3, 4, 3, 4, 3, 4]
with StimServer('localhost', port=4218) as stim_server:
    stim_server.start(verbose=True)
    for ev in events:
        stim_server.add_trigger(ev)

# in the stimulation computer(s)
stim_client = StimClient('localhost', port=4218)
trig = stim_client.get_trigger(timeout=0.2)
```

Code Snippet 3: StimServer and StimClient.

3.2.5 Unit testing

Unit testing is an essential component of software development. Unit tests are sanity checks to ensure that new changes in the repository maintain the integrity of the code base. Typically, unit tests need to be run often and thus they must be as efficient as possible while covering a high percentage of the repository. Therefore, a smaller dataset of 300 MB is designated for the unit tests. While a real script for scientific analysis may use all the sensor channels, a unit test would use only a subset of them. This makes them memory and time efficient.

Continuous integration of tests is performed using Travis CI[6] for every pull request on github. The code is compatible with three different versions of Python: Python2.6, Python2.7 and Python3.4. In addition, the Jenkins buildbot[7] builds the master branch at regular intervals to ensure integrity of code. In case of a failed build, developers are sent a notification by email. Developers use the `nosetests` package to run tests locally on their machines. The `doctest` searches for code snippets inside the comments section and runs them to ensure that the examples are still up to date.

Tests should consume minimal time so that they can be run whenever new changes are introduced. The `nosetimer` package can be used for profiling the tests. The `coverage` package gives a report of code coverage by the tests for each module separately along with lines which are not covered by the tests. The ambition is to have at least 80% code coverage for the entire repository. For every new commit to the repository, `coveralls` reports the increase or decrease in code coverage and sends an email to the GitHub repository.

For testing the realtime examples, it is necessary to have a mock server which can read in chunks of data from a `fiff` file and simulate the real MEG acquisition

[6] https://travis-ci.org/
[7] http://jenkins-ci.org/

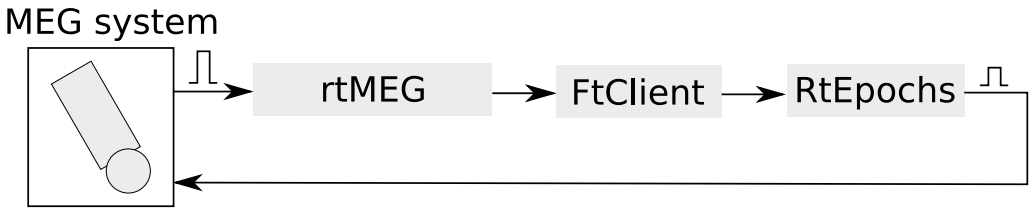

Figure 7: Measuring delay in the real-time system.

system. This was achieved at two levels. First, the `MockRtClient` is used to test the integrity of `RtClient`. Second, the FieldTrip connector is tested by modifying C code in the FieldTrip proxy program so that it can simulate data from a file by reading in `Fiff tags`. The playback speed of the data simulation can be controlled with a time delay specified by a `-speed` parameter. Ultimately, the MNE-Python tests spawn a new process to run the FieldTrip connector, acquires the data using `FtClient`, checks for integrity of the data and kills the process.

Sanity checks to ensure integrity of the data are performed using `assert` statements as shown in Code Snippet 4.

```
# data is read using the offline API from the file
# rt_data is read using MockRtClient -> RtEpochs

assert_true(rt_data.shape == data.shape)  # check dimensions
assert_array_equal(rt_data, data)  # check data
```

Code Snippet 4: Examples of unit tests.

3.3 Measuring real-time delay

To measure the delay due to the MNE real-time, the trigger channel on the MEG channel was activated with an inter-stimulus interval of 0.819 s. The MEG system and thus the FieldTrip buffer length was set to 28 ms in one case and to 100 ms in the other. In MNE-Python, epoching was done by selecting a time interval of -10ms to 20ms and the query size was set to correspond to 100 ms. As soon as an epoch was detected, a trigger code (different from the one received) was written to the trigger channel through the computer's parallel port. The data were saved to a *fiff* file for a 3 minute recording. Next, the triggers were extracted offline and the delay was measured. Figure 7 shows the procedure to measure the delay as a schematic diagram.

3.4 Stimuli and MEG experiments

Two healthy volunteers (one male and one female; both 24 years old) with normal hearing participated in the study. The two subjects will be referred to as S1 and S2. The sound levels and stimuli pitch difference on both ears were adjusted until the participants were able to detect the stimuli in the attended stream reliably without

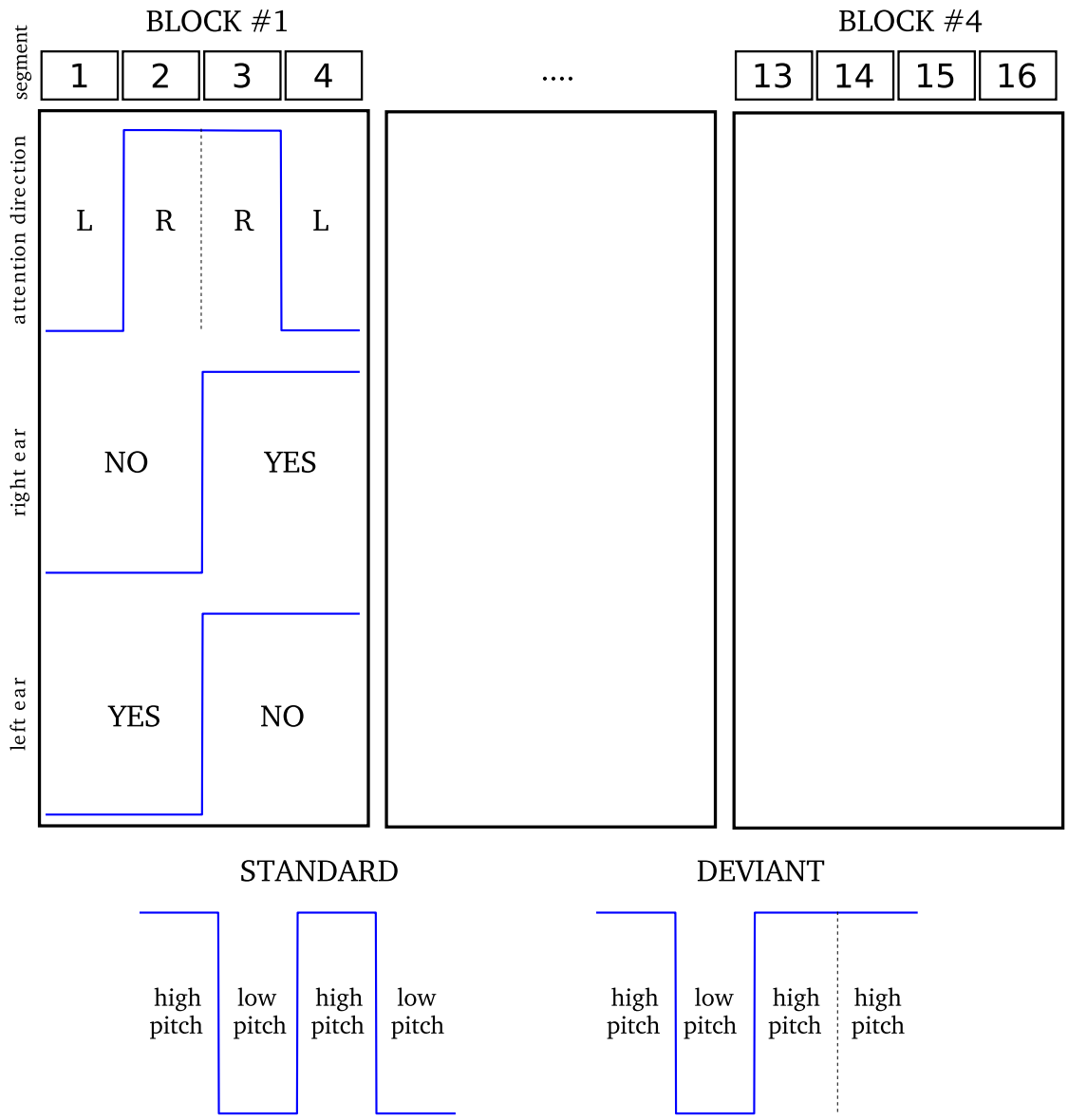

Figure 8: The design of the experiment.

getting distracted by the stimuli in the unattended stream. They were then asked to attend to one of the streams with their eyes open and fixated at a point straight in front of them. The entire experiment consisted of 16 segments and each segment contained approximately 100 stimuli on either stream. The experiment was divided into four blocks. In each block, two segments contained 'yes' on the left ear and the two others 'yes' on the right ear. The direction of attention was maintained in the left ear for two segments and on the right ear for the other two segments. Figure 8 illustrates the paradigm of the experiment.

The sound were recorded from two spatially separated speakers by a dummy head (Modular System MK2, Cortex Electronic) in a room of comparable size to

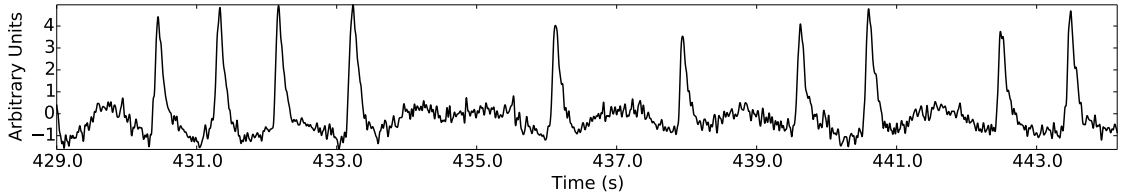

Figure 9: The time course of the ICA component corresponding to eyeblinks.

the MEG shielded room and then played back by earphones as two simultaneous streams. The Psychopy toolbox in Python (Peirce, 2007) was used for presenting the stimuli. The streams consisted of female 'yes' and male 'no' sounds, each in two different pitches: 'Yes' low pitch, 'Yes' high pitch, 'No' low pitch and 'No' high pitch. Audacity software[8] was used to shift the low pitch sound by 13% to 20% (different for different subjects) to obtain the high pitch sound. The low and high pitch sounds alternated in both the streams as standard stimuli. The deviant was a high pitch in a sequence of high–low–high–low pitch sounds, and it is presented with a probability of 10%. The stimulus onset asynchrony between the two streams was constant at 500 ms. The reason for choosing the 500-ms interval was that the P300 response from one stream would not contaminate the N100 and P300 responses to the other stream.

3.5 Data analysis

The raw data were first processed using the Maxfilter program (Version 2.2.10, Elekta Oy., Helsinki, Finland) which applies SSS transformation (see Section 2.6.2).

Eyeblink rejection: To reject eyeblinks, Independent Component Analysis (ICA) was performed on the unfiltered MEG data using the FastICA algorithm (Hyvärinen, 1999) implemented in MNE-Python. Using ICA, the signals in the MEG sensors can be expressed as a linear combination of the independent signal components. The independent signal components are also known as the *sources* and the linear combination is known as the *mixing matrix*. The original MEG signals can be recovered by the inverted mixing matrix (also known as the *unmixing matrix*) and multiplying it with the sources. To decontaminate eyeblinks in the MEG signal, we simply zero out the components corresponding to the eyeblinks in the mixing matrix and then recover the cleaned signals. This procedure (explained in more detail below) will subtract out the dynamics due to the eyeblinks in the signal.

The FastICA algorithm is based on using the fourth moment kurtosis, to maximize the non-gaussianity which is a measure of statistical independence of the signals. To maximize computational efficiency, a fixed-point iterative scheme is used which is much faster than traditional gradient-descent methods.

Figure 9 shows the time course of the source (for one subject) which contains eyeblinks. To find this component, an automatic procedure, which is implemented in the

[8] http://audacity.sourceforge.net/

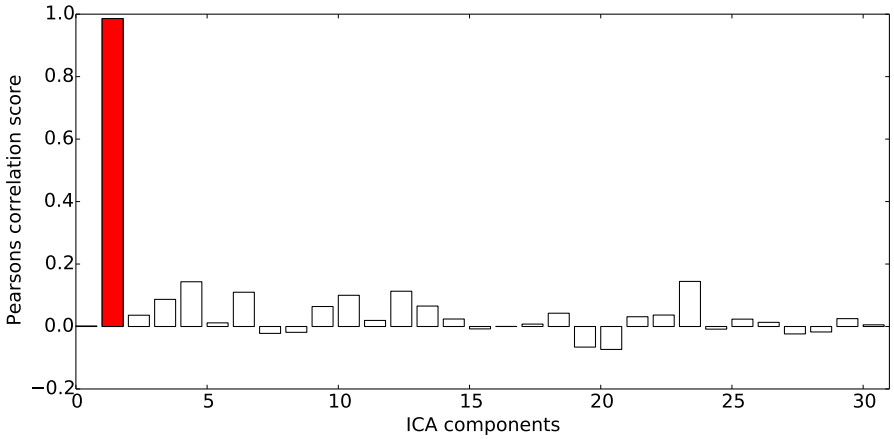

Figure 10: Correlation coefficient of the ICA components. The outlier (shown in red) is detected using adaptive z-scoring.

MNE-Python package, was followed. First, the signal components which accounted for upto 95% of the explained variance were retained and the rest were rejected. This ensured that only important signal components were retained and noisy components were removed. Next, a frontal MEG channel where the eyeblinks contaminated the signal the most was manually chosen. Pearson's correlation coefficient for this channel and each of the retained components were computed (Figure 10). To find the outlier component, an iterative method was used where in each iteration, the z-score (obtained by subtracting the mean and dividing by the standard deviation) of the correlations with the selected components was compared with a preselected threshold (3.0). Components which exceed the threshold were not considered for the next iteration. This was repeated until there were no components whose z-score of the correlation coefficient exceeds the threshold.

Next, the sources corresponding to the eyeblinks were isolated and visualized in the MEG sensor topography (Figure 11). The topographic plot shows that the component affects the frontal channels closest to the eyes the most. This confirms that the selected component indeed corresponds to eyeblinks.

Components corresponding to eye movements were not removed explicitly. Eye tracking can be used to remove epochs contaminated with eye movements. However, eye movements may not be a concern in an eventual application with vegetative state patients as most of them are not capable of moving their eyes.

Epoch extraction: The cleaned data were then filtered between 1.0 Hz and 30.0 Hz using MNE-Python. Bad epochs were rejected using a peak-to-peak threshold of $4 * 10^{-10}$ T/m for magnetometers and $4 * 10^{-12}$ T for gradiometers. Epochs were extracted as time windows starting -200 ms before the stimulus onset and lasting until 800 ms after the stimulus onset. For decoding analysis, the epochs were downsampled by a factor of 8, which reduced the learning time for the classifiers which can be beneficial for online decoding.

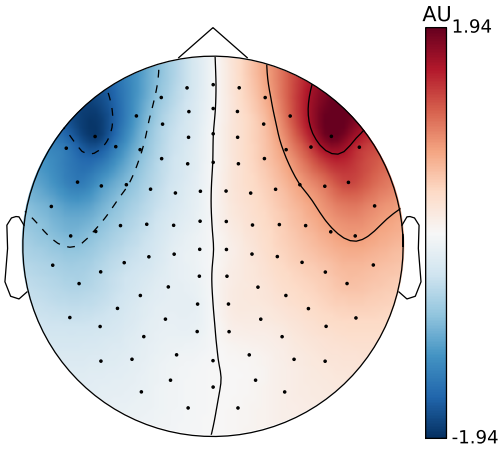

Figure 11: The unmixing matrix for the EOG source plotted on the MEG sensor topography and interpolated.

Source modelling: The MEG signal reflects the net effect of two kinds of currents: primary currents, which are mainly due to postsynaptic potentials (see Section 2.1), and volume currents, which are due to the interaction of the primary current with a conductive medium. Various methods for estimating the neuronal sources based on MEG signals exist. In dipole modelling, it is assumed that the activity can be represented by a current dipole, and the aim is to find the location (in the 3-dimensional space), orientation and magnitude of these dipoles. Numerical methods exist for dipole modelling. Dipole modelling works well for focal sources which can be represented by a single dipole but these methods are not suitable for distributed sources.

To overcome this limitation, distributed source models (MNE, dSPM *etc.*) assume a grid of dipoles in a volume or the surface and find a solution that best explains the measured MEG signal. It allows incorporating anatomical constraints into the model.

In MNE-Python, the implementation of these distributed methods rely on computation of two operators: the *forward operator* and the *inverse operator*. A forward operator describes how the dipolar sources are represented in the sensor signals. The inverse operator describes how the sensor signals can be transformed to obtain the dipole magnitudes. The forward operator computed using the Boundary Element Model (BEM) requires definition of the source space (locations where the dipoles will be placed), the segmented cortical surfaces (outer skull, inner skull and outer skin surface) to model the volume currents and the coordinate transformation between the MEG device and the MRI device. These steps require the cortical surface which can be reconstructed from the MRI scan using the Freesurfer software package[9] (Dale and Sereno, 1993; Dale et al., 1999). Once the cortical surface is found, the segmentation of the surfaces can be performed using the watershed algorithm (Ségonne et al.,

[9] http://freesurfer.net/

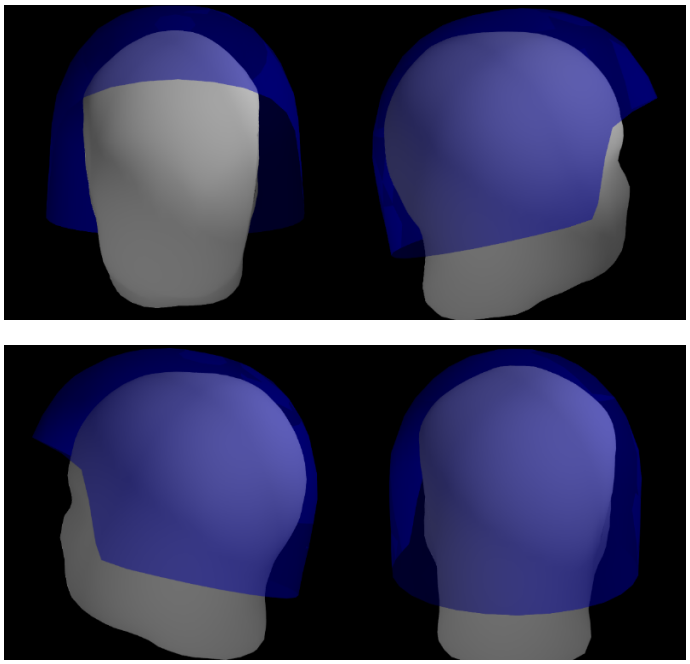

Figure 12: The quality of coregistration between MEG and MRI.

2004) implemented in Freesurfer. The process of aligning the MEG and the MRI coordinates is known as coregistration. The inverse operator can be obtained from the forward operator and the noise covariance matrix. The noise covariance matrix informs about the noise levels and spatial correlations between sensor channels. Once the inverse operator is available, it can be applied on the MEG data to obtain the dipole magnitudes.

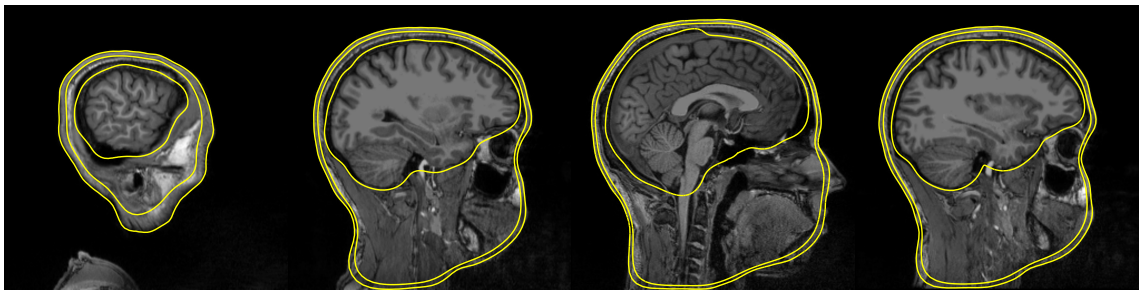

Figure 13: The surfaces of the three compartments of the BEM model along the sagittal axis for 4 MRI slices spaced equally along the axis.

It is critical to ensure that each of these steps in the analysis pipeline are performed correctly. The quality of coregistration can be checked by plotting the surface of the helmet (transformed to the MRI coordinates using the coordinate transformations) and head together (Figure 12). Ideally, the surfaces should not intersect and the

head should be aligned and spaced equally from all direction inside the helmet as in the MEG device. The quality of the BEM segmentation can be ensured by browsing through the MRI 2D slices overlayed with the curve obtained when the segmented surface intersects with that slice. Ideally, the overlayed line should align with the boundary of the inner skull, outer skill and outer skin surface which is visible in the MRI slice. The BEM segmentation is inspected along each of the axes: sagittal, coronal and axial. Figure 13 shows an example for one subject.

4 Results and Discussion

First, the results related to the development of the software packages will be described. Then, I will delve into the case study where the real-time API is applied to decoding attention.

4.1 Realtime interface

4.1.1 Unit tests

Unit tests are important to ensure the integrity of the software package (described in more detail in Section 3.2.5). The aim is to write unit tests which cover as much of the codebase as possible. Table 4 shows the coverage of the `realtime` and `decoding` modules in MNE-Python. Local tests were done on a 64-bit machine with a 2.4-GHz Intel i7 processor running Linux Mint 16. The results indicate that the `RtClient` needs improved coverage. This is due to the fact that the `RtClient` can only work when the MNE real-time server is available. This can be solved in the future by creating a `MockRtServer` which simulates the MNE real-time server.

Module	Coverage (Local)	Coverage (Travis)
mne.realtime	100%	100%
mne.realtime.client	18%	18%
mne.realtime.epochs	85%	82%
mne.realtime.fieldtrip_client	72%	68%
mne.realtime.mockclient	91%	91%
mne.realtime.stim_server_client	88%	88%
mne.decoding	100%	100%
mne.decoding.classifier	91%	91%
mne.decoding.mixin	71%	71%

Table 3: Coverage of unit tests on python2.7.

Table 4 shows the time consumed and the peak memory requirements for each of the tests. All tests consume very little memory which makes it suitable to run on older computers. The `test_mockclient.py` is slightly slow because it terminates only when no stimulus is detected for a pre-determined duration (2 s by default).

4.1.2 Delay due to MNE real-time

Timing is crucial in real-time MEG measurements, especially those involving feedback. Typically, in these experiments, the data are acquired and analyzed and the subject is presented with feedback depending on the analysis (before the next stimulus is presented). However, if the latency of the real-time system is too large, the feedback may cross over to subsequent stimulus resulting in incorrect feedback. Therefore, all components of the real-time pipeline should have as low a latency as possible. Delay was measured using the procedure outlined in Section 3.3.

Test file	No. of tests	Time consumed (s)	Peak memory (MB)
test_mockclient.py	2	2.18	1
test_fieldtrip_client.py	1	1.31	1
test_stim_client_server.py	1	1.01	1
test_classifier.py	4	0.50	1

Table 4: Timing and memory of unit tests on the local machine in python2.7.

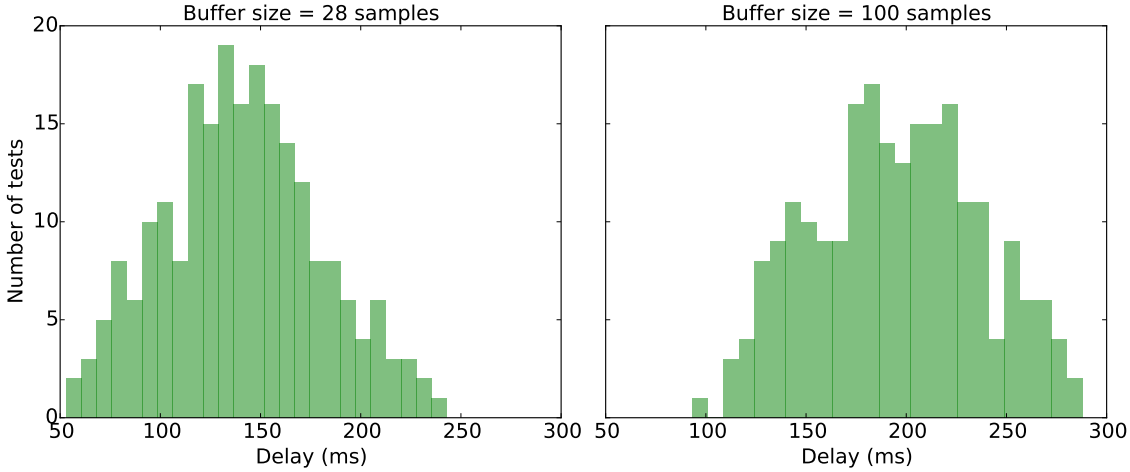

Figure 14: The delay due to MNE real time with a buffer size of 100 ms in the FieldTrip buffer and a query length of 100 ms in MNE-Python.

Figure 14 shows the distribution of the delays in individual trials as a histogram. The total delay is due to the buffer lengths on the FieldTrip buffer, the query length in MNE-Python, the epoch length and network delays over the entire loop. It should be noted that the FieldTrip buffer and the MNE-Python analysis script are running asynchronously. Therefore both are adding, on an average, a delay equal to their length depending on where the trigger occurred with respect to the two buffers. Thus, an average delay of 128 ms is expected in the first case and 200 ms in the second case. This delay is within the limits of most experiments involving real-time feedback.

4.2 Stimuli

Auditory stimuli delay: There can be a delay between the trigger signals and the actual stimuli due to various reasons – the onset timing of the acoustic stimuli, processing delays in the computers, and the propagation of the sound wave from the loudspeaker to the ear. To measure the delay due the ear, an artificial ear was used to imitate the inner ear. An oscilloscope was connected to the trigger channels and to the output of the artificial ear in two separate channels. First, a tone pip was delivered several times and the delay was measured between the trigger and the tone

pip.

Stimuli	delay (ms)
Tone pip	45
Yes (high pitch)	56
Yes (low pitch)	52
No (high pitch)	55
No (low pitch)	56

Table 5: Delays associated with the auditory stimuli.

In Table 5, we can observe that this delay was found to be 45 ms. Next, each of the stimuli types was delivered repeatedly and the delay between the trigger channel and the average acoustic response of the stimuli was noted. This delay was typically 10 ms more than the delay associated with the tone pip. This is because of a silent time period that was added to the beginning of each stimuli to avoid an abrupt onset of the stimuli. The calculated delay was taken into account in all the following analysis.

4.3 Classification analysis

In the experiments described subsequently, a linear SVM with regularization coefficient C = 1.0 was used. The signal in all the sensor channels was downsampled by a factor of 8 to obtain a sampling frequency of 125 Hz. To avoid aliasing, the signal was low-pass filtered with a cut-off frequency of 30 Hz, lower than the Nyquist frequency. This helped decrease the dimensionality of the feature space and reduce the learning time of the classifier.

4.3.1 Offline analysis

First, I will describe analysis performed offline after collection of the data. Next, we will describe the setup for online decoding of attention. Analyzing the data offline is important as it allows us to optimize the experiment and the decoding pipeline for real-time use.

Training on all channels and all time points using all available epochs: The epochs were first randomly shuffled and then split it into a training set and a test set in the ratio of 80:20. The classifier was trained on the training set and its performance was evaluated on the test set. A decoding accuracy of 86% was achieved when averaged over 5 such splits. This type of splitting procedure may actually result in biased estimates because the data are correlated across time. If the test epoch was selected from an epoch close to a training epoch, the classifier may find it easier to classify because of the presence of a training epoch very close to the test epoch. However, other schemes for cross-validation (*e.g.* stratified K-fold cross-validation) did not impact the results significantly.

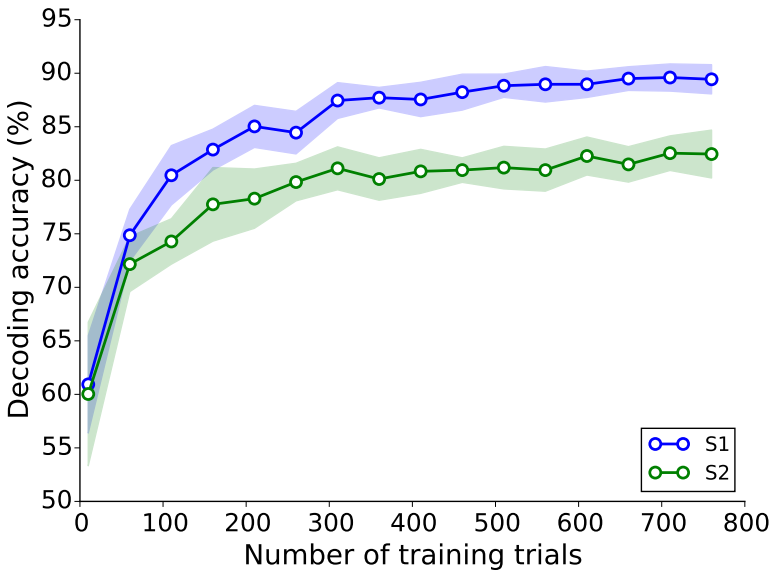

Figure 15: Decoding accuracy against number of training trials.

Increasing training data: For a BCI application, it is important to know how much training data are sufficient for satisfactory decoding performance. The number of epochs used for training was increased from 10 to 760 in steps of 50, and the performance of the trained SVM model was tested on a test set consisting of 300 epochs. We found that the decoding accuracy saturates after around 200 trials which corresponds to about is around 4 minutes of measurement time.

Training on one experimental block and testing on another: Even though we were able to decode with a high accuracy when the training and test splits were random, we are more interested in the case where the training and test splits are in fact not random. In a realistic real-time experiment, the training of the data would be on one experimental block and the testing would be on another. This will additionally test that the classifier is not learning correlations in the data but can actually predict test data which are well separated in time from the training data. We trained our classifier on the first two segments of the first block (Section 3.4) and tested it on the first two segments of the following block. A decoding accuracy of 83.9% was achieved indicating that the results are generalizable across time.

Effect of filter cut-off frequencies on decoding accuracy: In Section 2.3, several studies are described which have linked the oscillatory dynamics in the brain to maintaining attention. This led us to investigate if certain frequency bands contributed more to the decoding accuracy compared to others. First, the lowpass cut-off frequency was varied in steps of 10.0 Hz (15.0, 25.0 and 35.0 Hz), keeping the highpass cut-off frequency fixed at 1.0 Hz. The different cut-off frequencies gave rise to almost equal decoding rates. However, something surprising emerged as we tried to lower the frequencies even further. It appeared that the decoding accuracy was not

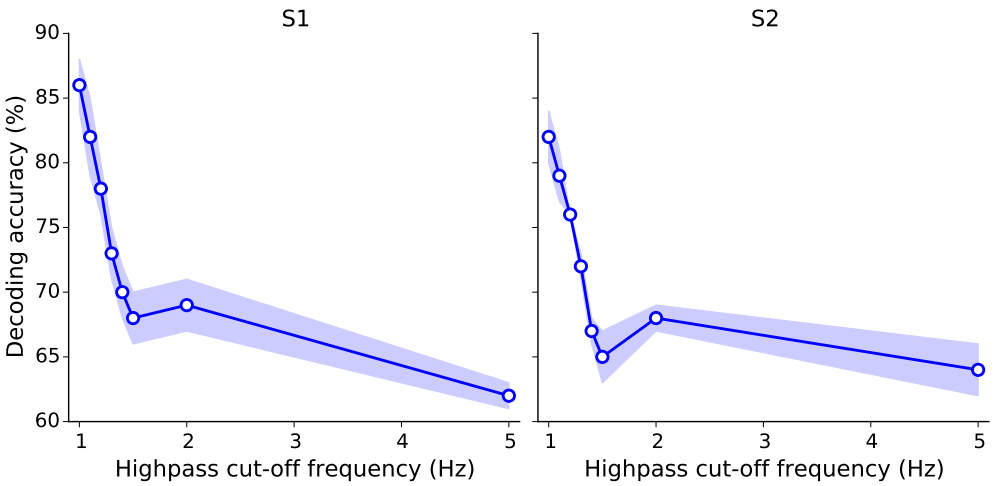

Figure 16: Decoding accuracy as a function of highpass cut-off frequency.

affected even if the lowpass cut-off frequency was as low as 2.0 Hz. In fact, a slight rise in the decoding accuracies was observed when the lowpass cut-off frequency was lowered to 2.0 Hz.

To investigate this phenomena further, the lowpass cut-off frequency was fixed at 30.0 Hz and the highpass cut-off frequency was varied from 1.0 Hz to 5.0 Hz. It can be observed that there is a steep drop in the decoding accuracy for the cut-off frequency in the range 1.0–1.5 Hz. Increasing the cut-off frequency further to 5.0 Hz led to slight decline in the decoding rate from 2.0 Hz.

SVM weights: The surprisingly low frequency at which information was decodable required further scrutiny. One hypothesis was that the decoding could simply be explained by low-frequency drifts in the MEG signal from one experiment block to the other. If the decoder was able to detect these changes in amplitude drifts, this would lead to a high decoding rate but it would not be physiologically meaningful. To rule out this possibility, the Support Vector Machine weights were computed. The SVM weights (Equation 7) form the normal vector to the separating hyperplane and therefore, each dimension in the input feature space is associated with its own weight. Since the input feature space is the set of all time points (from 50 ms to 700 ms) concatenated over all channels, we have SVM weights for all the channels in this time interval.

First, the MEG signal was filtered to a frequency band of 1.0–1.5 Hz (transition bandwidth = 0.5 Hz). Next, an SVM classifier was trained over all the channels and time points (ranging from 50 ms to 700 ms). Finally, the weights were extracted and arranged them according to the channel and time point they belong to. This resulted in Figure 17 which visualizes the weights on the MEG sensor topography. Thus, we have a spatiotemporal view of how the classifier discriminates between the two classes. We find that the weights in the temporal channels show the highest

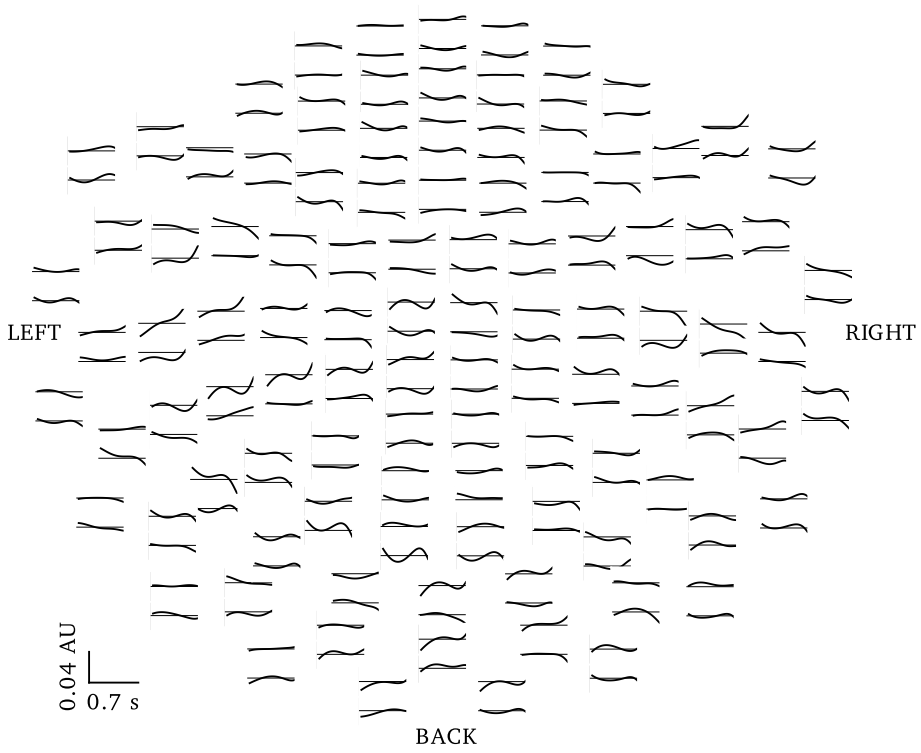

Figure 17: SVM weights on the sensor topography (subject S1).

fluctuating amplitude indicating that these channels are important in the decoding. This means that the classifier was indeed not relying on low-frequency drifts (which would be present at all sensor channels) but was rather discriminating using the features in the temporal channels closest to the auditory cortex. The fluctuations in the SVM weights are around 1.0–1.5 Hz corresponding to the filtered frequency band.

Control analysis: The results on control analysis are reported only on subject S1. Decoding on even and odd samples of the same stimuli yielded chance-level decoding accuracy of $44.0 \pm 8.0\%$ for the attended right (yes on right ear) condition. This indicates that the classifier was not merely decoding statistical differences in noise between the two conditions. Further, separately decoding attention when yes was presented to the left ear ($83.0 \pm 3.0\%$) or when yes was presented to the right ear ($86.0 \pm 2.0\%$) retained the high classification rate. This means that the ear to which a particular stimuli is presented does not impact the results.

4.3.2 Time-resolved decoding

In traditional MEG analysis, the epochs are often averaged and these averages are compared between two conditions. However, the average response does not take into account the noise levels in the sensors and is not sensitive to differences in patterns of activity in individual trials. Therefore, time-resolved decoding (Ramkumar et al.,

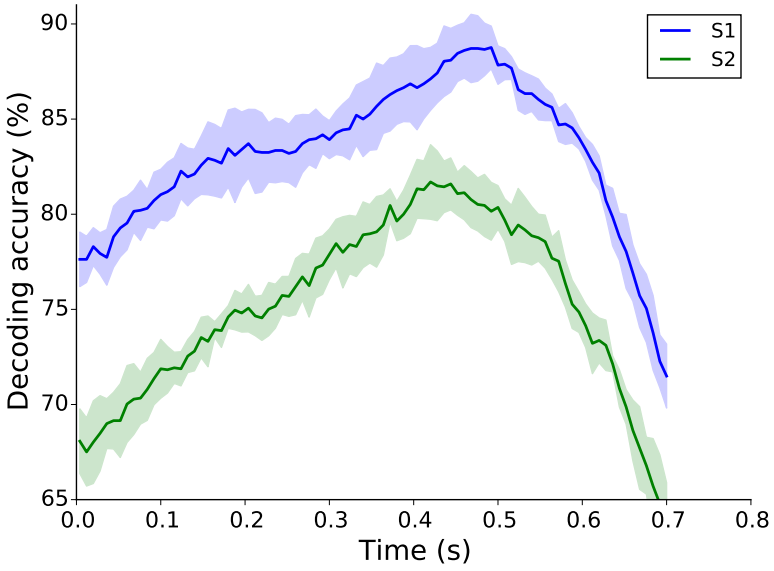

Figure 18: Time-resolved decoding with all gradiometer channels and a small time window of 16 ms sliding in steps of 8 ms.

2013) has been suggested as a more sensitive alternative to evoked response analysis. In this method, the decoding rate as a function of time is treated as an indicator of the task-related information content.

In this analysis, the same frequency band (1.0–1.5 Hz) as in the previous section was used and the decoding accuracy was computed over a small time window in the decimated signal concatenated across all channels. A window size of 2 samples (= 2*1000 Hz/125 Hz = 16 ms for a decimation factor of 8) was selected and this window was moved in steps of 1 sample (8 ms). This results in a time-resolved decoding curve (Figure 18). The decoding accuracy peaks at around 600 ms which is approximately the time instant at which the sound in the other stream starts to be processed in the brain. It is worth noting that a classification accuracy of almost 90% is achieved (for subject S1) using only two time points over all the gradiometer channels, *i.e.* a total of 408 features.

4.3.3 Sensor-by-sensor

Now that we have explored the temporal aspect of the decoding accuracy, we are interested in exploring the spatial aspect. This was done by training a classifier on all time points (from 0 ms to 700 ms) but on only one channel at a time. The decoding accuracy is plotted on the sensor topography and interpolated using bilinear interpolation.

This approach is complementary to visualizing the SVM weights on the sensor topography. While the former is critical to understanding which features (both temporally and spatially) are most important during decoding, this approach tells us which sensors are the most important when the temporal dimension is ignored.

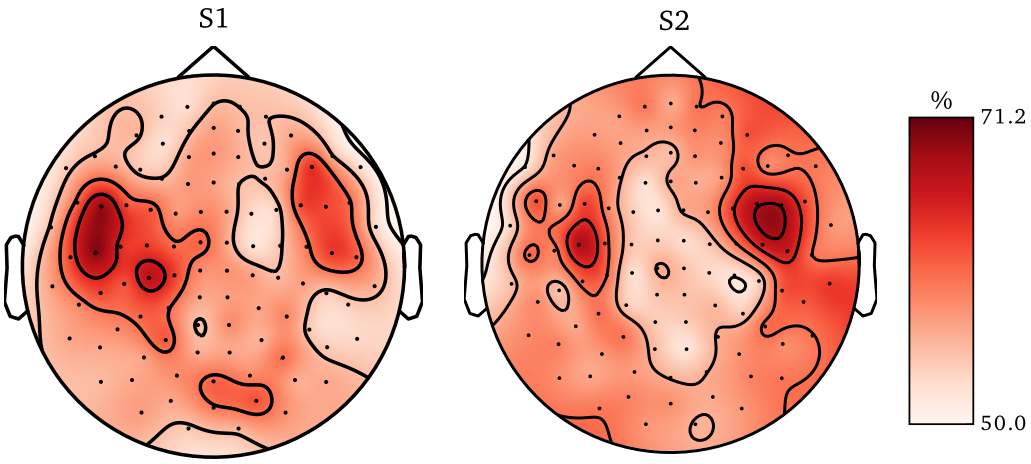

Figure 19: The decoding accuracies when decoding is performed sensor-by-sensor. A smooth colormap is obtained by interpolating the decoding accuracies using bilinear interpolation.

Moreover, SVM weights were computed using the training data and thus indicate the importance of the features during the *training* phase. However, sensor-by-sensor decoding evaluates the performance of classifiers trained using individual channels on the *test* dataset.

Figure 19 shows that the hotspots of information content in the attention decoding tasks are near the left temporal and right temporal channels. Therefore, the temporal channels not only provide important features for decoding but also contain the maximum information for the attention decoding task. Of course, in a real-time application, we could use this result to select only a subset of the channels (in the temporal areas) and achieve a reasonably high decoding accuracy using fewer features.

4.3.4 Simulation of real-time decoding

The proxy program that implements the FieldTrip buffer (Section 3.1.1) has the capability to simulate the real-time experiment by reading in chunks of data from the stored `fiff` file. The recorded `fiff` file was first modified to store only 4 segments consisting of the first two segments in the first block and the first two segments in the second block. The short length of the new `fiff` file helps in faster debugging of the real-time analysis script. The classifier was trained on the first block and tested on the second block (similar to the experiment on decoding across experimental blocks described above). If the filtering was performed offline and the rest of the analysis was performed in real-time, a decoding accuracy of 83.9% (the same as offline analysis) was achieved.

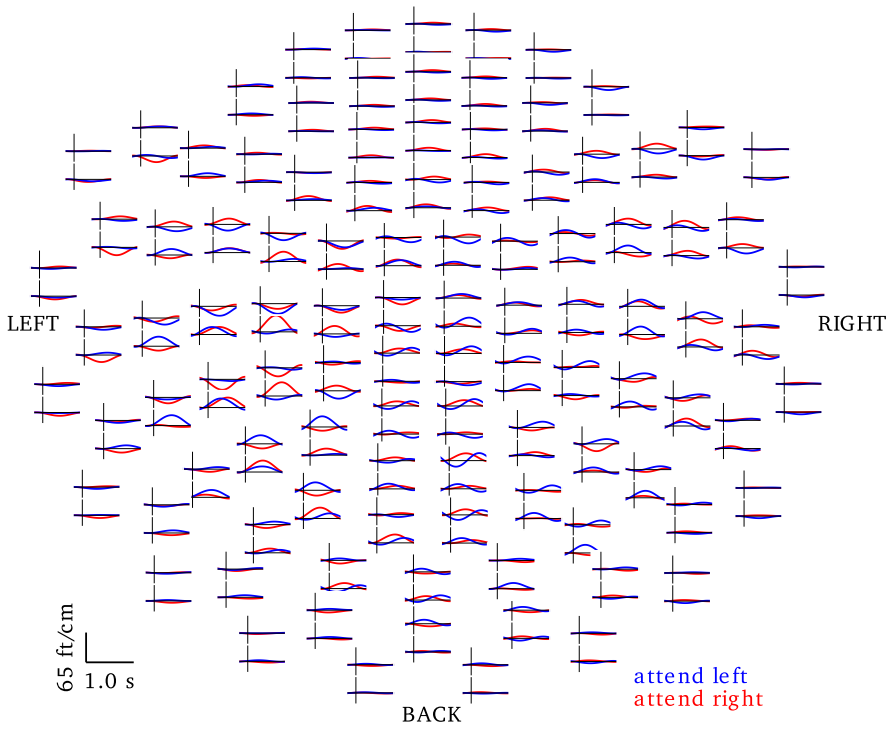

Figure 20: The evoked response filtered to a frequency band between 1.0–1.5 Hz (subject S1).

4.4 Evoked responses

In addition to classification analysis, the evoked responses to the attended left *vs*. attended right categories can be studied. The evoked responses were obtained by averaging the signal for each channel across different trials (also known as epochs). Figure 20 shows these responses plotted on the sensor topography for signals that have been filtered to a frequency band of 1.0–1.5 Hz. There was a large difference in the evoked responses over the temporal channels which is consistent with the observations in Figure 17 and 19, and could indicate a lateralization of the underlying neural activity at the frequency of stimulation. Choosing the frequency of stimulation to focus the attention might indicate that the subject was shadowing each stimuli in the attended stream. However, a large difference in the evoked response need not necessarily mean that the decoding accuracy will be high in those channels. This is because the decoding accuracy depends on how discriminative the features are in both the training and the test set. A simple example of two features that have a large difference in their mean across the two conditions but are not discriminative enough is when the standard deviation in the two conditions is also large. Therefore, differences in evoked responses is necessary, but not sufficient for satisfactory classification performance.

4.5 Source estimates

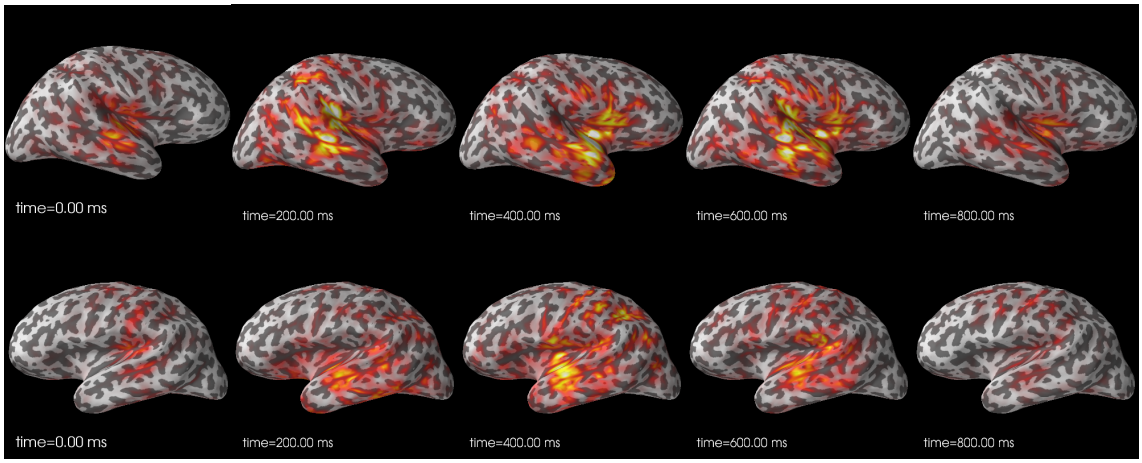

Figure 21: Source estimates computed with dSPM on the attended left condition (subject S1). The top row shows the evolution of the source estimates over time for the right hemisphere. The bottom row shows this evolution for the left hemisphere.

Until now, all the analysis described was in the sensor space. Since the differences were mainly concentrated over the temporal channels, they are probably due to differences in the auditory cortex. However, source modelling must be performed to accurately localize the activations in the underlying neural sources. The purpose of source modelling is to predict the sources in the brain that can explain the signals in sensor space. In Section 3.5, the various steps in source modelling was described.

Here, a distributed source modelling approach known as dynamic Statistical Parameter Mapping (dSPM) was used for source modelling. DSPM can be considered an extension of the classical Minimum Norm Estimate (MNE) method where the euclidean norm of the current sources is minimized. DSPM is essentially the MNE inverse solution of the signal divided by the MNE of the noise.

Figure 21 shows the activations when the subject was attending towards the left auditory stream (no sound). The top row shows the evolution of the source activations in the right hemisphere whereas the bottom row shows that in the left hemisphere. The estimates peak in the temporal lobes in both the hemispheres. The activations start from a baseline zero when the stimuli is presented, rise up after the presentation of the sound and return back to baseline level at around 800 ms. Thus, as speculated in Section 4.4, there is indeed a lateralization of the underlying neural activity due to attention.

5 Summary

In this thesis, a real-time analysis pipeline for MEG analysis was proposed, supporting online machine learning. The latency for of this platform was around 100 ms which is within the limits of the allowable delays in most MEG experiments. The pipeline allows connecting to the FieldTrip buffer seamlessly. The contributions to the open source package MNE-Python that were required to enable this pipeline were tested by unit tests with a coverage of more than 85% for 6 out of 8 modules. The tests were fast (less than 1.5 s each) and consumed minimal memory (around 1 MB).

Experiments were conducted on healthy human subjects to decode the direction of auditory attention. Offline analysis of these data was performed using machine learning methods. It was found that the frequency band 1.0–1.5 Hz contains was most informative for the decoding task. This result was robust irrespective of how the training and test splits were chosen, and decoding could even be performed by training on one experimental block and testing on another. It is noteworthy that a decoding accuracy around 90% was achieved when this narrow frequency band was selected. This potentially means that the brain is tracking the stimuli at the same frequency as they are being presented. Moreover, the decoding accuracy peaked when the stimuli in the other stream appeared, with a delay of around 500 ms with the attended stream. Just selecting 16 ms of data retained the same information about attention – as a 500-ms epoch after the presentation of the auditory stimuli. Of course, one could simply retain all the time points but the benefit of selecting a subset of time points is that it results in faster decoding during real-time experiments. A time–frequency picture of information processing emerges where the frequency band in the range of 1.0 to 1.5 Hz is most important for decoding and the time points around 500 ms in this frequency band are most critical.

Analyzing the SVM weights revealed that mostly the temporal channels were used for the decoding. A space–time–frequency picture emerged as the decoding was performed sensor-by-sensor. All these analyses reinforce the fact that decoding is due to physiological differences in the temporal channels. In fact, we are able to zone into three potentially important factors that drive attention in our dichotic listening experiment. First, the brain is locking into the frequency of stimulation. This approach is orthogonal to studies (Jääskeläinen and Ahveninen, 2014) where the stimuli delivery adapts according to the brain state. Here, the stimulus delivery was fixed beforehand but the brain adapted to it. Second, it tries to maximize the differences in the neural responses to these two conditions only around the temporal channels which are known to be responsible for processing auditory information. Thirdly, the suppression efforts are maximum when the stimuli in the other stream is presented.

Finally, the developed real-time backbone was used as a platform for the decoding of attention and the results achieved using offline analysis were replicated in pseudo real-time mode. Future work will explore the effects of using steady-state responses as features to the decoding algorithms and extend this pipeline to a fully-fledged BCI application.

References

Ahlfors, S. and R. J. Ilmoniemi (1989). Magnetometer position indicator for multichannel MEG. In *Advances in biomagnetism*, pp. 693–696. Springer.

Baars, B. J. (1993). *A Cognitive Theory of Consciousness*. Cambridge University Press.

Bonnefond, M. and O. Jensen (2012). Alpha oscillations serve to protect working memory maintenance against anticipated distracters. *Current Biology 22*(20), 1969–1974.

Cherry, E. C. (1953). Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America 25*(5), 975–979.

Cohen, D. (1968). Magnetoencephalography: evidence of magnetic fields produced by alpha-rhythm currents. *Science 161*(3843), 784–786.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Machine Learning 20*(3), 273–297.

Cox, D. D. and R. L. Savoy (2003). Functional magnetic resonance imaging (fMRI) "brain reading": detecting and classifying distributed patterns of fMRI activity in human visual cortex. *Neuroimage 19*(2), 261–270.

Cruse, D., S. Chennu, C. Chatelle, T. A. Bekinschtein, D. Fernández-Espejo, J. D. Pickard, S. Laureys, and A. M. Owen (2011). Bedside detection of awareness in the vegetative state: a cohort study. *The Lancet 378*(9809), 2088–2094.

Dale, A. M., B. Fischl, and M. I. Sereno (1999). Cortical surface-based analysis: I. Segmentation and surface reconstruction. *Neuroimage 9*(2), 179–194.

Dale, A. M. and M. I. Sereno (1993). Improved localizadon of cortical activity by combining EEG and MEG with MRI cortical surface reconstruction: a linear approach. *Journal of cognitive neuroscience 5*(2), 162–176.

Faugeras, O., F. Clément, R. Deriche, R. Keriven, T. Papadopoulo, J. Roberts, T. Viéville, F. Devernay, J. Gomes, G. Hermosillo, et al. (1999). The inverse EEG and MEG problems: The adjoint state approach I: The continuous case.

Gao, H., M. Ouyang, D. Zhang, and B. Hong (2011). An auditory brain-computer interface using virtual sound field. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pp. 4568–4571. IEEE.

Goldfine, A. M., J. C. Bardin, Q. Noirhomme, J. J. Fins, N. D. Schiff, and J. D. Victor (2013). Reanalysis of "Bedside detection of awareness in the vegetative state: a cohort study.". *The Lancet 381*(9863), 289.

Golestani, A. and R. Gras (2014). Can we predict the unpredictable? *Scientific Reports 4*.

Gramfort, A., M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, et al. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience 7*.

Hämäläinen, M. S. and R. Ilmoniemi (1994). Interpreting magnetic fields of the brain: minimum norm estimates. *Medical & Biological Engineering & Computing 32*(1), 35–42.

Hansen, P., M. Kringelbach, and R. Salmelin (2010). *MEG: An introduction to methods*. Oxford University Press.

Haxby, J. V., M. I. Gobbini, M. L. Furey, A. Ishai, J. L. Schouten, and P. Pietrini (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science 293*(5539), 2425–2430.

Hill, K. T. and L. M. Miller (2009). Auditory attentional control and selection during cocktail party listening. *Cerebral Cortex 20*(3), 583–590.

Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks 10*(3), 626–634.

Jääskeläinen, I. P. (2012). Introduction to Cognitive Neuroscience. http://goo.gl/5IYpdg. Accessed: 2015-03-02.

Jääskeläinen, I. P. and J. Ahveninen (2014). Auditory-cortex short-term plasticity induced by selective attention. *Neural plasticity 2014*.

Jensen, O., M. Bonnefond, and R. VanRullen (2012). An oscillatory mechanism for prioritizing salient unattended stimuli. *Trends in Cognitive Sciences 16*(4), 200–206.

Kamitani, Y. and F. Tong (2005). Decoding the visual and subjective contents of the human brain. *Nature Neuroscience 8*(5), 679–685.

Kay, K. N., T. Naselaris, R. J. Prenger, and J. L. Gallant (2008). Identifying natural images from human brain activity. *Nature 452*(7185), 352–355.

King, J.-R., A. Gramfort, A. Schurger, L. Naccache, and S. Dehaene (2014). Two distinct dynamic modes subtend the detection of unexpected sounds. *PloS One 9*(1), e85791.

Koch, C. and N. Tsuchiya (2007). Attention and consciousness: two distinct brain processes. *Trends in Cognitive Sciences 11*(1), 16–22.

Kolossa, A., T. Fingscheidt, K. Wessel, and B. Kopp (2012). A model-based approach to trial-by-trial p300 amplitude fluctuations. *Frontiers in Human Neuroscience 6*.

Kriegeskorte, N., R. Goebel, and P. Bandettini (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the United States of America 103*(10), 3863–3868.

Maddox, R. K., W. Cheung, and A. K. Lee (2012). Selective attention in an overcrowded auditory scene: Implications for auditory-based brain-computer interface design. *The Journal of the Acoustical Society of America 132*(5), EL385–EL390.

Mars, R. B., S. Debener, T. E. Gladwin, L. M. Harrison, P. Haggard, J. C. Rothwell, and S. Bestmann (2008). Trial-by-trial fluctuations in the event-related electroencephalogram reflect dynamic changes in the degree of surprise. *The Journal of Neuroscience 28*(47), 12539–12545.

Matsumoto, Y., S. Makino, K. Mori, and T. M. Rutkowski (2013). Classifying P300 responses to vowel stimuli for auditory brain-computer interface. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pp. 1–5.

Mitchell, T. M., S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just (2008). Predicting human brain activity associated with the meanings of nouns. *Science 320*(5880), 1191–1195.

Näätänen, R., A. W. Gaillard, and S. Mäntysalo (1978). Early selective-attention effect on evoked potential reinterpreted. *Acta psychologica 42*(4), 313–329.

Naci, L., M. M. Monti, D. Cruse, A. Kübler, B. Sorger, R. Goebel, B. Kotchoubey, and A. M. Owen (2012). Brain–computer interfaces for communication with nonresponsive patients. *Annals of Neurology 72*(3), 312–323.

Nambu, I., M. Ebisawa, M. Kogure, S. Yano, H. Hokari, and Y. Wada (2013). Estimating the intended sound direction of the user: toward an auditory brain-computer interface using out-of-head sound localization. *PloS One 8*(2), e57174.

Naselaris, T., K. N. Kay, S. Nishimoto, and J. L. Gallant (2011). Encoding and decoding in fMRI. *Neuroimage 56*(2), 400–410.

Oostenveld, R., P. Fries, E. Maris, and J.-M. Schoffelen (2010). FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. *Computational Intelligence and Neuroscience 2011*.

Owen, A. M., M. R. Coleman, M. Boly, M. H. Davis, S. Laureys, and J. D. Pickard (2006). Detecting awareness in the vegetative state. *Science 313*(5792), 1402–1402.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research 12*, 2825–2830.

Ramkumar, P., M. Jas, S. Pannasch, R. Hari, and L. Parkkonen (2013). Feature-specific information processing precedes concerted activation in human visual cortex. *The Journal of Neuroscience 33*(18), 7691–7699.

Schreuder, M., B. Blankertz, and M. Tangermann (2010). A new auditory multi-class brain-computer interface paradigm: spatial hearing as an informative cue. *PloS One 5*(4), e9813.

Ségonne, F., A. Dale, E. Busa, M. Glessner, D. Salat, H. Hahn, and B. Fischl (2004). A hybrid approach to the skull stripping problem in MRI. *Neuroimage 22*(3), 1060–1075.

Squires, K. C., C. Wickens, N. K. Squires, and E. Donchin (1976). The effect of stimulus sequence on the waveform of the cortical event-related potential. *Science 193*(4258), 1142–1146.

Sudre, G., L. Parkkonen, E. Bock, S. Baillet, W. Wang, and D. J. Weber (2011). rtMEG: a real-time software interface for magnetoencephalography. *Computational Intelligence and Neuroscience 2011*, 11.

Taulu, S., M. Kajola, and J. Simola (2004). Suppression of interference and artifacts by the signal space separation method. *Brain topography 16*(4), 269–275.

Thomson, D. J. (2007). Jackknifing multitaper spectrum estimates. *Signal Processing Magazine, IEEE 24*(4), 20–30.

Tzovara, A., A. O. Rossetti, L. Spierer, J. Grivel, M. M. Murray, M. Oddo, and M. De Lucia (2013). Progression of auditory discrimination based on neural decoding predicts awakening from coma. *Brain 136*(1), 81–89.

Wessberg, J., C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. Nicolelis (2000). Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature 408*(6810), 361–365.

Wilsch, A., M. J. Henry, B. Herrmann, B. Maess, and J. Obleser (2014). Alpha oscillatory dynamics index temporal expectation benefits in working memory. *Cerebral Cortex*.

Wolpaw, J. R., N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan (2002). Brain–computer interfaces for communication and control. *Clinical Neurophysiology 113*(6), 767–791.

Zion Golumbic, E. M., N. Ding, S. Bickel, P. Lakatos, C. A. Schevon, G. M. McKhann, R. R. Goodman, R. Emerson, A. D. Mehta, J. Z. Simon, et al. (2013). Mechanisms underlying selective neuronal tracking of attended speech at a "cocktail party". *Neuron 77*(5), 980–991.